Modelli Matematici per la Biologia – Esercitazioni a.a. 2006-2007

Dott. Simone Zuccher

2 giugno 2007

Sono qui raccolte le esercitazioni di Modelli Matematici per la Biologia svolte per il Corso di Laurea in Matematica Applicata presso l'Università degli Studi di Verona nell'anno accademico 2006-2007.

Nota. Queste pagine potrebbero contenere degli errori: chi li trova è pregato di segnalarli all'autore (zuccher@sci.univr.it).

Indice

1 Simulazione al calcolatore di vari modelli discreti del tipo $x_{n+1} = f(x_n)$ e $x_{n+1} = f(x_n, x_{n-1})$

1.1 Esercizio

Sia $x_0 \in (0, \pi)$ e si consideri la successione definita da

$$x_{k+1} = x_k + \sin x_k.$$

1. provare che $x_k \in (0, \pi)$ per ogni $k \in \mathbb{N}$;
2. provare che (x_k) cresce;
3. calcolare il limite di (x_k) per $k \to \infty$.

1.1.1 Risoluzione

1. Si noti che $x_{k+1} = f(x_k)$ con $f(x) = x + \sin x$. $f(x)$ è strettamente crescente su $(0, \pi)$ in quanto $f'(x) = 1 + \cos x$ e $0 < 1 + \cos x < 2$ $\forall x \in (0, \pi)$, ovvero $f'(x) > 0$ $\forall x \in (0, \pi)$. Essendo $f(0) = 0$, $f(\pi) = \pi$, e la funzione strettamente crescente, allora $x_{k+1} = f(x_k) \in (0, \pi)$.
2. Essendo $x_{k+1} - x_k = \sin x_k > 0$ $\forall x_k \in (0, \pi)$, la successione è crescente.
3. Per $k \to \infty$ si ha l'equazione $x = x + \sin x$, che è soddisfatta per $x = 0$, $x = \pi$ e $x = +\infty$. Di questi, l'unico limite possibile è $x = \pi$ in quanto la successione è crescente e si parte da $x_0 \in (0, \pi)$. Un altro modo è di osservare che $f'(0) = 2 > 0$ e pertanto l'origine è un punto unito instabile, mentre $f'(\pi) = 0$ e quindi è stabile.

Tutte le caratteristiche di (x_k) sopra menzionare sono verificabili graficamente utilizzando il file `esempio1.m` per GNU Octave riportato in appendice A. Esso richiede come input N (numero di iterazioni – quindi nel caso $k \to +\infty$ si deve scegliere N opportunamente elevato) e il valore iniziale x_0 (chiamato `s(1)` in `esempio1.m`).

In figura 1 è riportata la storia temporale e il *cobwebbing* (procedura a "zigo-zago" ottenuta partendo da x_0, calcolando $x_1 = f(x_0)$ tramite la f, riportando il valore x_1 sulla bisettrice del primo e terzo quadrante, quindi calcolando $x_2 = f(x_1)$ tramite la f, e così via) ottenuti per $N = 15$ e $x_0 = 0.1$. Si noti il raggiungimento veloce del valore asintotico $x_n = \pi$.

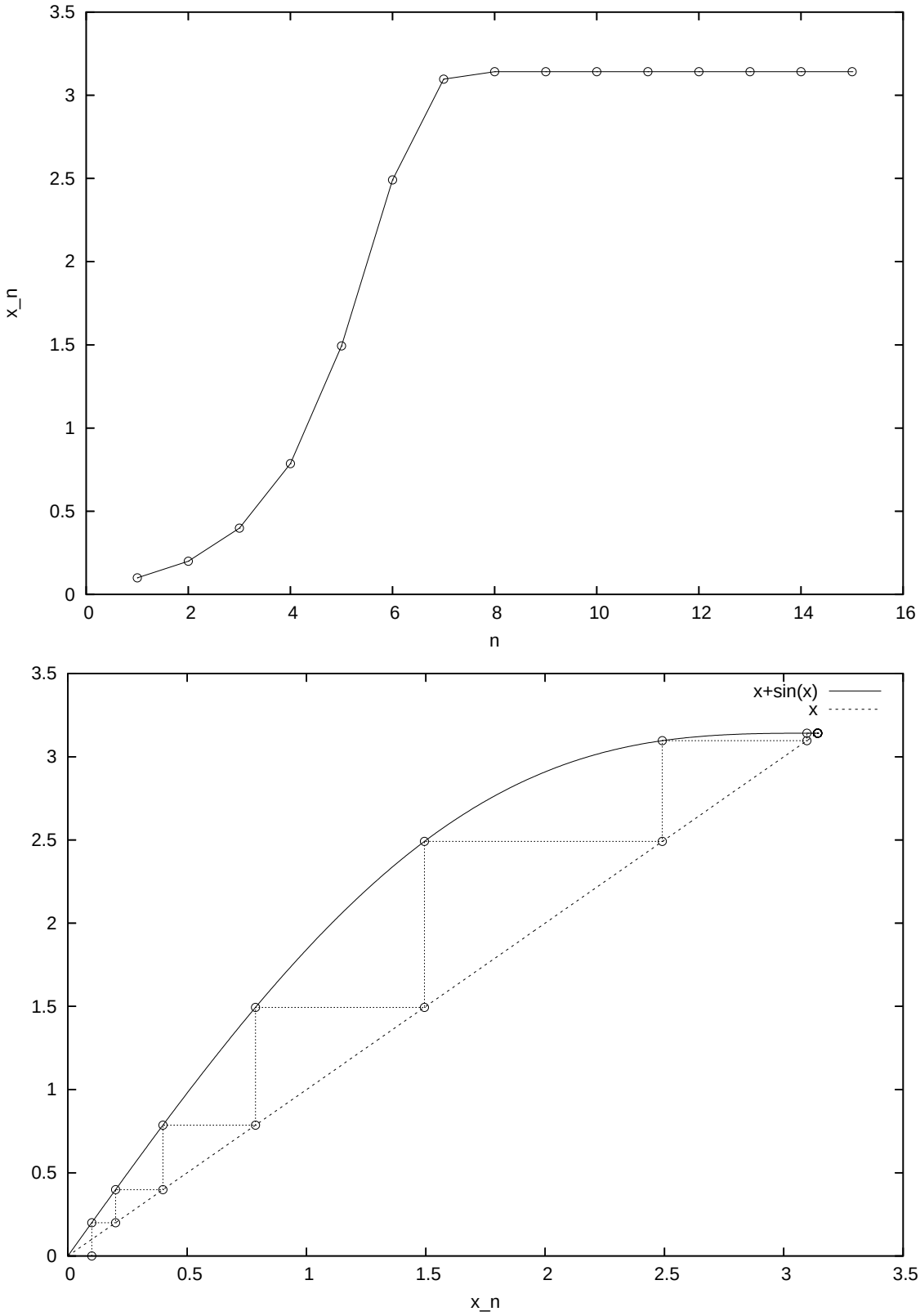


Figura 1: Storia temporale x_n e studio dei punti uniti, $N = 15$, $x_0 = 0.1$.

1.2 Esercizio

Data la successione (x_k) definita da

$$x_0 = a, \qquad x_{k+1} = \max\left\{\frac{1}{4}, x_k^2\right\}$$

dire se esiste, al variare di $a \in \mathbb{R}$, il limite di (x_k) per $k \to \infty$.

1.2.1 Risoluzione

In questo caso $x_{k+1} = f(x_k)$ con $f(x) = \max\left\{\frac{1}{4}, x_k^2\right\}$. Questa funzione è decrescente per $x < -1/2$, costante e pari a $1/4$ per $-1/2 \leq x \leq 1/2$ e crescente per $x > 1/2$. I punti uniti si trovano risolvendo l'equazione $x = \max\left\{\frac{1}{4}, x^2\right\}$ che ha come soluzioni $x = 1/4$ e $x = 1$. Anche $x = +\infty$ è un possibile limite della successione (x_k) per $k \to +\infty$ in quanto soddisfa l'equazione dei punti uniti. La funzione data ha derivata nulla in $x = 1/4$, che pertanto è stabile, e derivata pari a $2 > 0$ in $x = 1$, che è pertanto instabile. Si verifica facilmente che il limite della successione è $x = 1$ se $x_0 = \pm 1$; il limite è $x = 1/4$ per $|x_0| < 1$, mentre il limite è $+\infty$ per $|x_0| > 1$.

`esempio2.m` può essere utilizzato per la soluzione grafica di questo esercizio. Esso richiede come input N (numero di iterazioni – quindi nel caso $k \to +\infty$ si deve scegliere N opportunamente elevato) e il valore iniziale x_0 (chiamato `s(1)` in `esempio2.m`).

In figura 2 sono riportati la storia temporale e il *cobwebbing*, ottenuti per $N = 7$ e $x_0 = -1$, dove si nota che la soluzione è attratta da $x = 1$. In figura 3 sono riportati la storia temporale e il *cobwebbing*, ottenuti per $N = 7$ e $x_0 = -0.7$, dove si nota che la soluzione è attratta da $x = 1/2$; infine, in figura 4 sono riportati la storia temporale e il *cobwebbing* ottenuti per $N = 7$ e $x_0 = -1.01$, dove si nota l'esplosione della soluzione.

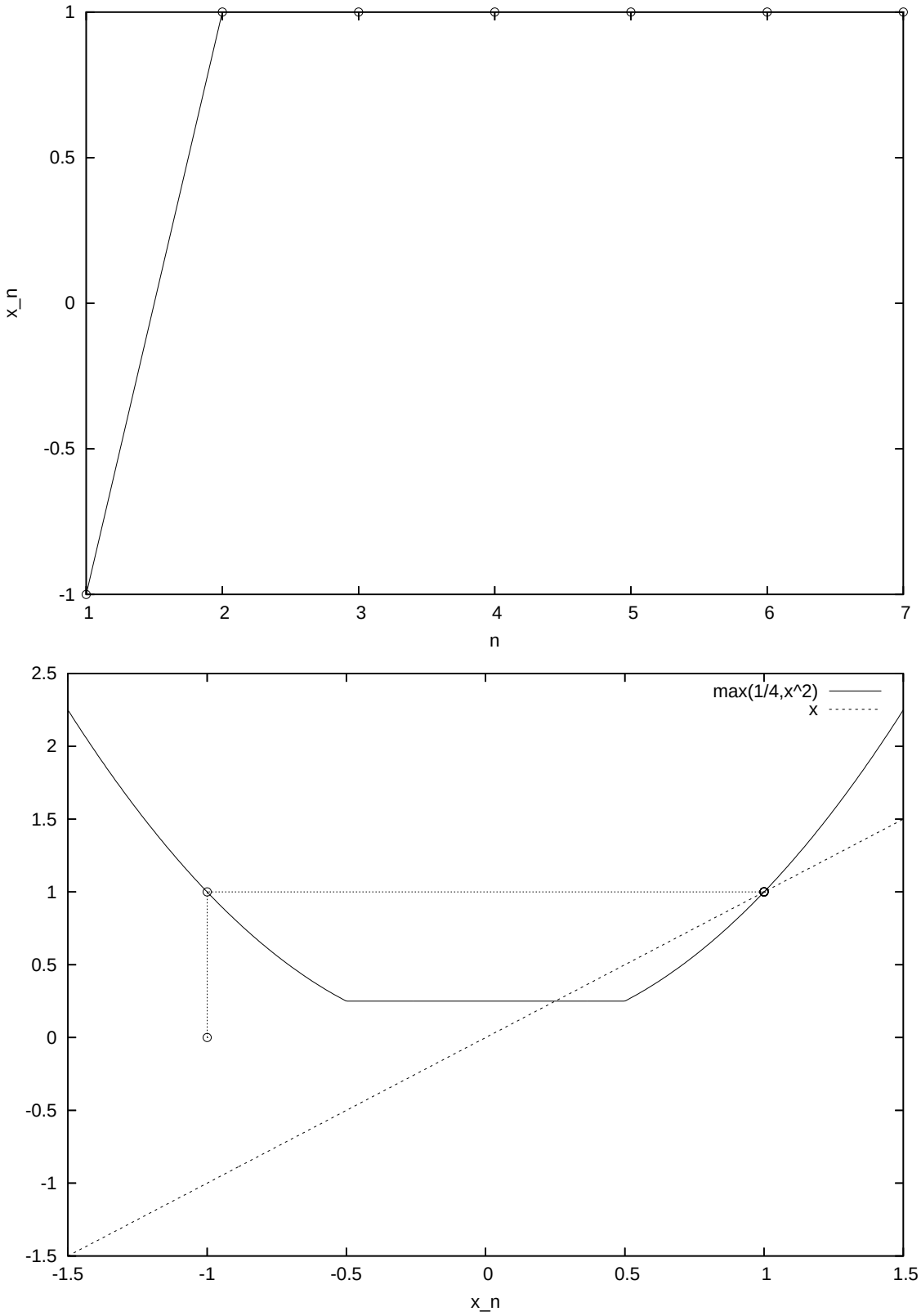


Figura 2: Storia temporale x_n e studio dei punti uniti, $N = 7$, $x_0 = -1$.

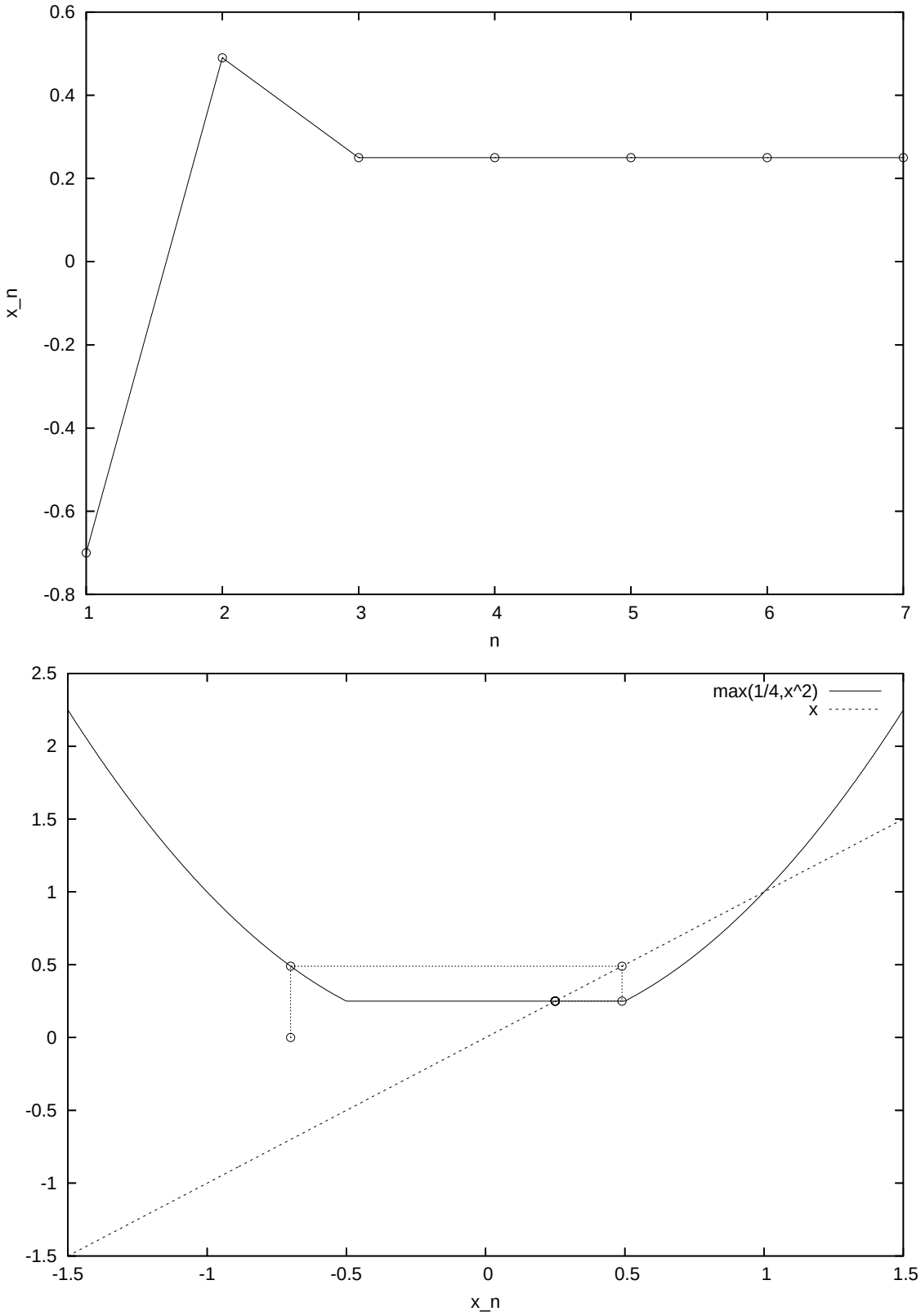


Figura 3: Storia temporale x_n e studio dei punti uniti, $N = 7$, $x_0 = -0.7$.

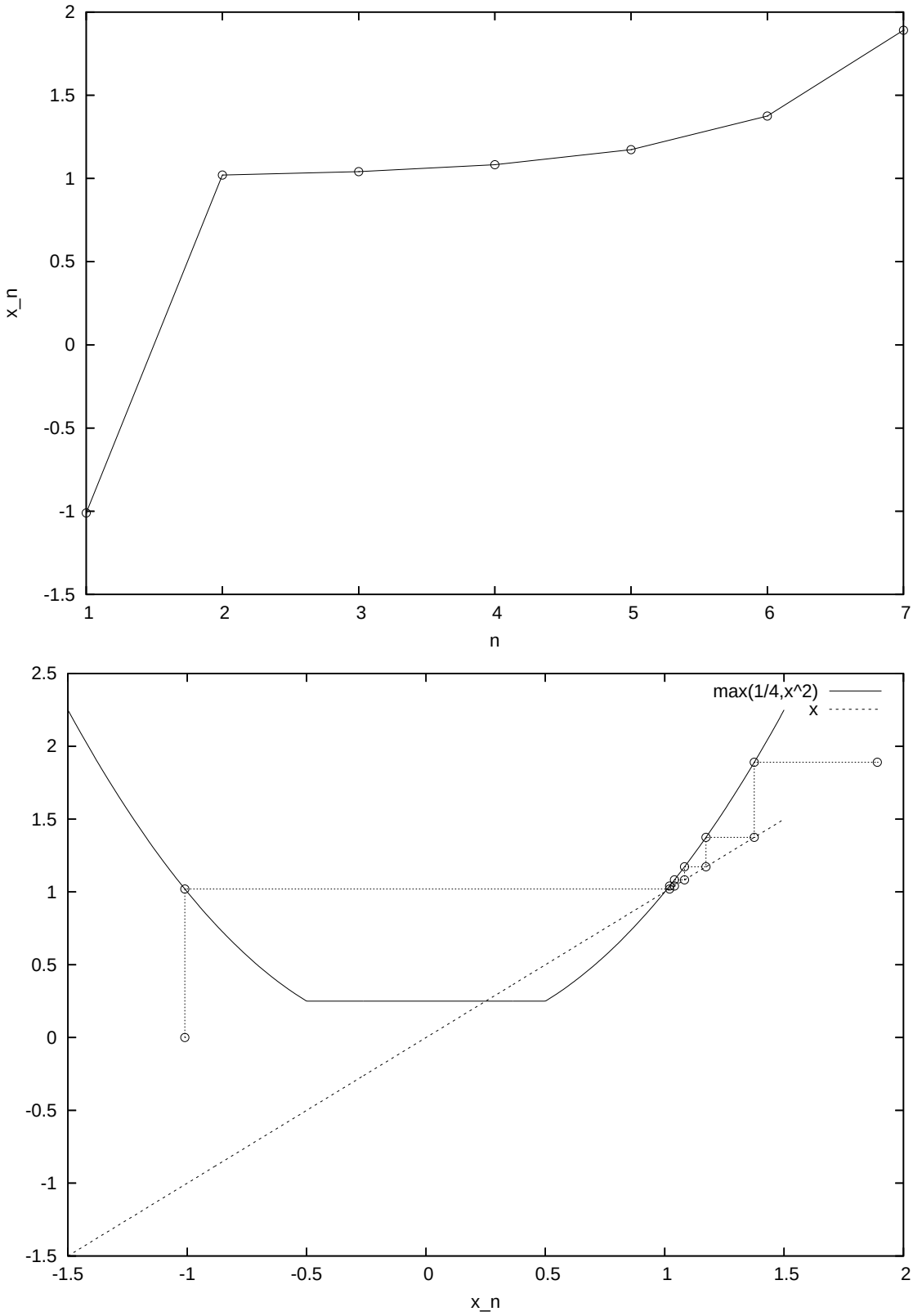


Figura 4: Storia temporale x_n e studio dei punti uniti, $N = 7$, $x_0 = -1.01$.

1.3 Esercizio

Calcolare il limite della successione definita da

$$x_0 = 1, \qquad x_{k+1} = \int_0^{x_k} e^{-t^2} dt.$$

1.3.1 Risoluzione

Si noti che $x_{k+1} = \frac{\sqrt{\pi}}{2}\,\mathrm{erf}(x_k)$, ovvero $f(x) = \frac{\sqrt{\pi}}{2}\,\mathrm{erf}(x)$. Essendo $f'(x) = e^{-x^2} > 0\ \forall x \in \mathbb{R}$, la funzione è sempre crescente ed essendo $x_2 < x_1$ la successione è decrescente. L'unico punto unito che soddisfa l'equazione $x = \frac{\sqrt{\pi}}{2}\,\mathrm{erf}(x)$ è $x = 0$, che risulta il limite della successione in quanto essa è decrescente e $x_0 = 1$.

Si osservi che l'analisi di stabilità del punto $x = 0$ porta a $f'(0) = 1$ ($f'(x) = e^{-x^2}$), per cui la determinazione della sua natura richiede l'uso delle derivate successive secondo quanto riportato in figura 5. Essendo $f''(x) = -2xe^{-x^2}$, si ha $f''(0) = 0$ e quindi bisogna analizzare la derivata terza che è $f'''(x) = 2e^{-x^2}(2x^2 - 1)$. Essendo $f'''(0) = -2 < 0$, $x = 0$ è un punto di equilibrio localmente asintoticamente stabile.

`esempio3.m` può essere utilizzato per la soluzione grafica di questo esercizio. Esso richiede come input solo N (numero di iterazioni) essendo il valore iniziale $x_0 = 1$ fissato.

In figura 6 sono riportati la storia temporale e il *cobwebbing*, ottenuti per $N = 500$. Si può notare la natura dell'origine, attrattiva se pur con una velocità di convergenza molto bassa.

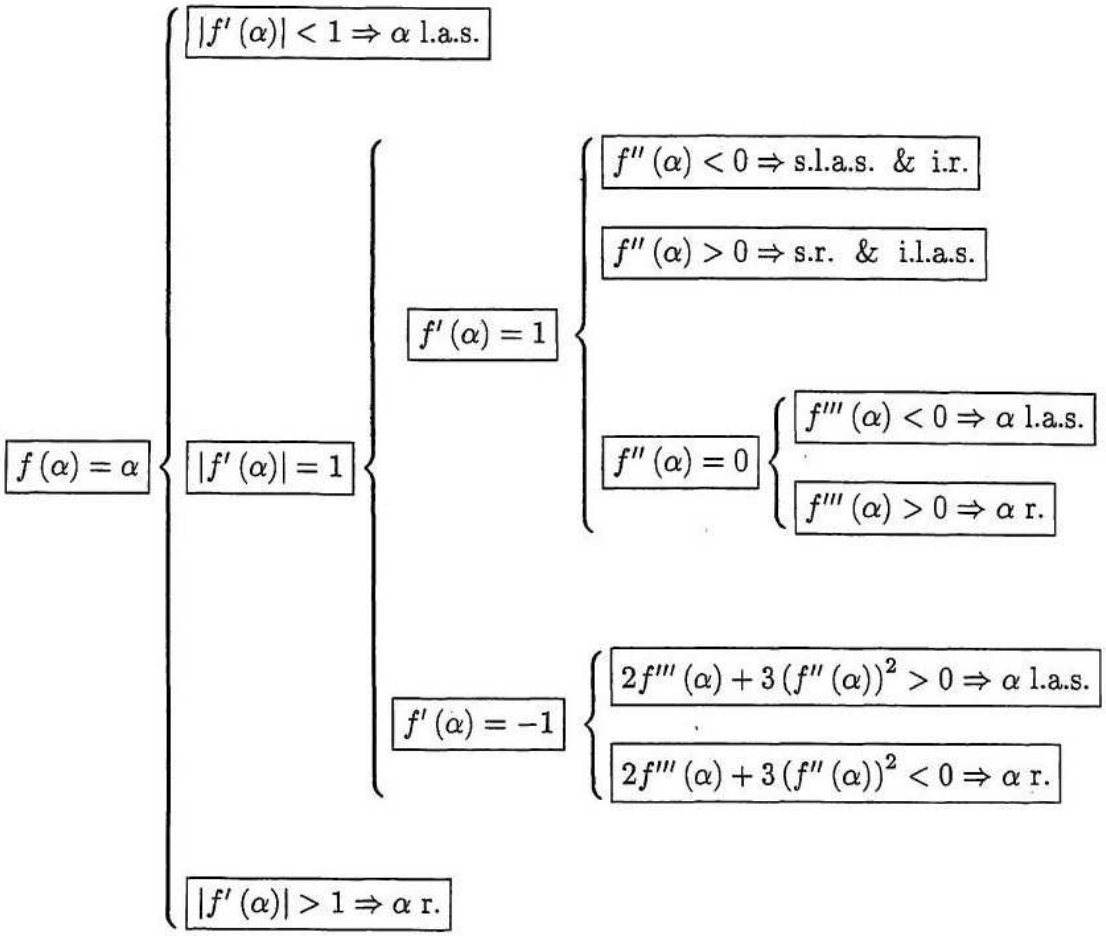
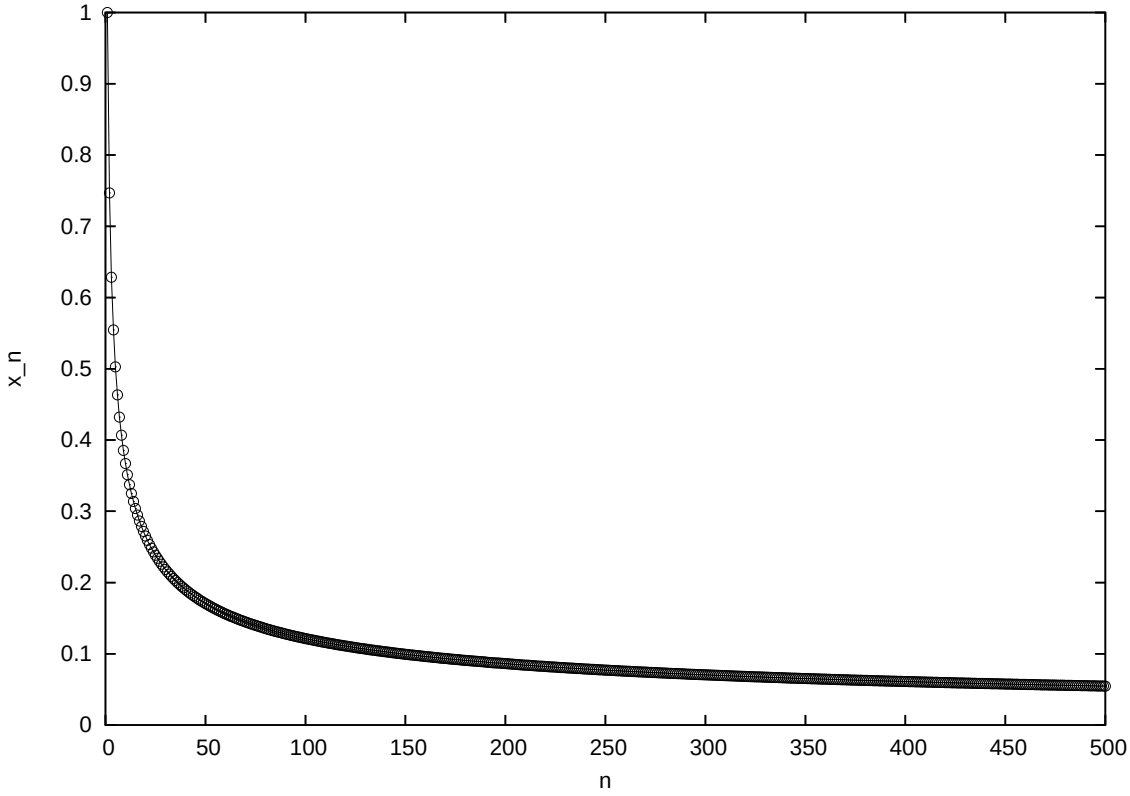


Tabella 3.1 Schema riassuntivo per lo studio della stabilità di un equilibrio α quando f è dotata di derivate. Legenda:

l.a.s. = localmente asintoticamente stabile
s.l.a.s. = superiormente localmente asintoticamente stabile
i.l.a.s. = inferiormente localmente asintoticamente stabile
r. = repulsivo
s.r. = superiormente repulsivo
i.r. = inferiormente repulsivo

Figura 5: Schema per la determinazione della natura dei punti di equilibrio per mappe del tipo $x_{n+1} = f(x_n)$.



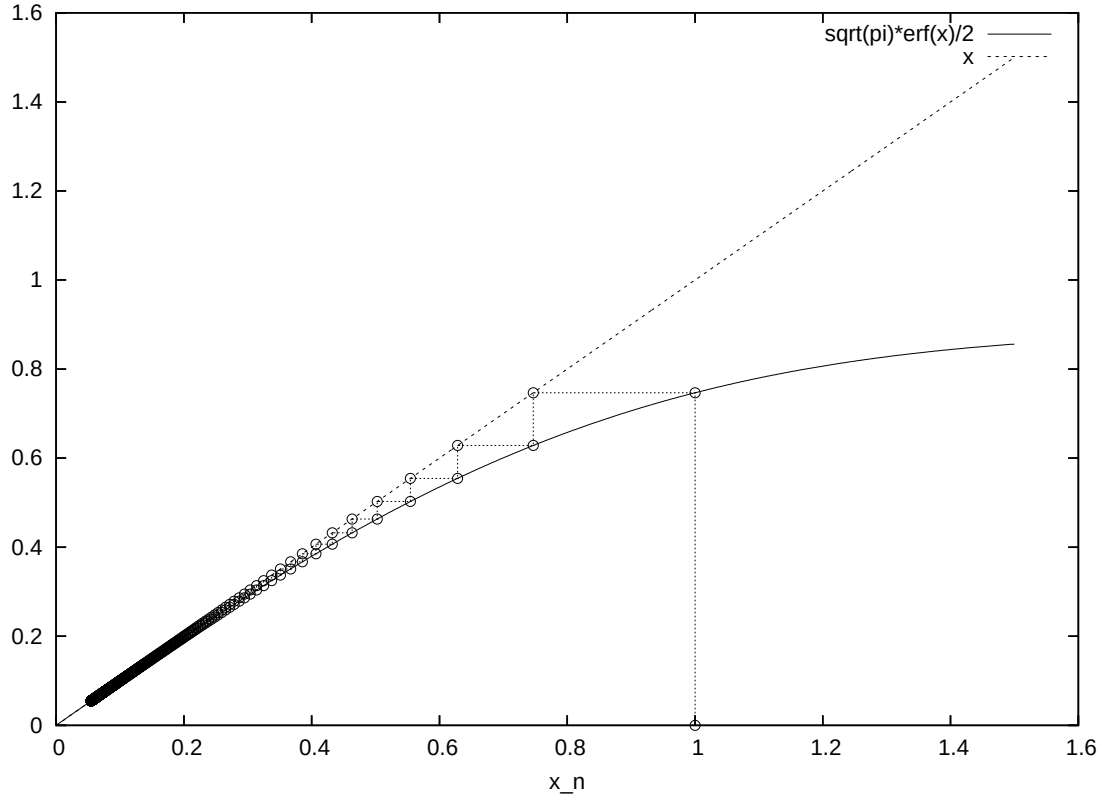


Figura 6: Storia temporale x_n e studio dei punti uniti, $N = 500$.

1.4 Esercizio

Si consideri l'equazione

$$x^k = \cos \frac{x}{k}.$$

1. provare che esiste un'unica soluzione $x_k > 0$;
2. provare che (x_k) rimane limitata;
3. calcolare il limite di (x_k) per $k \to \infty$.

1.4.1 Risoluzione

1. Disegnando il grafico di $y = x^k$ e $y = \cos \frac{x}{k}$, come riportato in figura 7, si osserva facilmente che le due funzioni si intersecano in un solo punto $x_k \in (0, 1)$ (si osservi che, per k pari, le intersezioni sono due, di cui una sola positiva).
2. A seguito del punto precedente, la successione stessa (x_k) rimane limitata tra 0 e 1.
3. Al tendere all'infinito di k la funzione $y = \cos \frac{x}{k}$ diventa sempre più "piatta" (costante) nell'intorno di $x = 1$ dove vale 1 in quanto l'argomento del coseno tende a zero. Siccome, per $k \to +\infty$, $x = 1$ soddisfa l'equazione $x^k = \cos \frac{x}{k}$, il limite della successione è 1.

`esempio4.m` può essere utilizzato per la soluzione grafica di questo esercizio. Esso richiede come input N (numero di iterazioni) e il valore iniziale di tentativo per la funzione che calcola la radice $x_k > 0$. Si consiglia di fornire un valore $0 < x_{\text{guess}} < 1$.

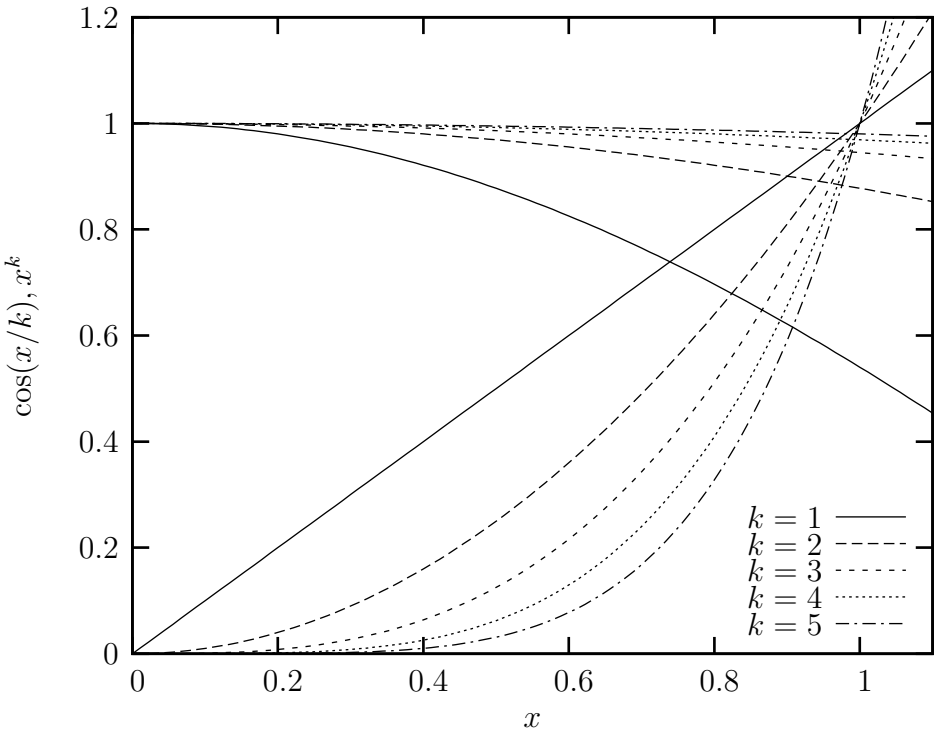


Figura 7: $y = \cos\frac{x}{k}$ e $y = x^k$ al variare di k. Si noti come il punto di intersezione sia per $0 < x < 1$, al limite è $x \to 1$ per $k \to \infty$.

1.5 Esercizio

Si consideri la successione definita da

$$x_0 = \lambda, \qquad x_{k+1} = \frac{x_k}{1 + x_k},$$

con $\lambda \geq 0$. Calcolare il limite di (x_k) per $k \to \infty$.

1.5.1 Risoluzione

Essendo $f(x) = \frac{x}{1+x}$ strettamente crescente per $x \geq 0$, essendo $x = 0$ il suo unico punto unito, ed essendo $x_2 < x_1$, la successione è strettamente decrescente ed ha come limite $x = 0$ $\forall x_0 \geq 0$. Alternativamente, si osservi che $\lim_{x\to 0^{\pm}} f'(x) = \lim_{x\to 0^{\pm}} 1/(x+1)^2 = 1^{\mp}$ e quindil'origine è stabile superiormente e instabile inferiormente. Una ulteriore alternativa era l'analisi classica secondo lo schema 5. Da $f'(x) = \frac{1}{(x+1)^2}$ si ottiene $f'(0) = 1$, da $f''(x) = -\frac{2}{(x+1)^3}$ segue $f''(0) = -2 < 0$, per cui il punto di quilibrio $x = 0$ è superiormente localmente asintoticamente stabile e inferiormente repulsivo. Qui interessa che sia stabile superiormente.

Si osservi che la successione data modellizza una legge di crescita di una popolazione con risorse limitate e tasso di natalità inversamente proporzionale alle dimensioni della popolazione, pertanto condannata all'estinzione.

`esempio5.m` può essere utilizzato per la soluzione grafica di questo esercizio. Esso richiede come input N (numero di iterazioni) e il valore iniziale $x_0 = \lambda$.

In figura 8 sono riportati la storia temporale e il *cobwebbing*, ottenuti per $N = 500$ e $x_0 = 0.9$. Si può notare la natura dell'origine, attrattiva se pur con una velocità di convergenza molto bassa.

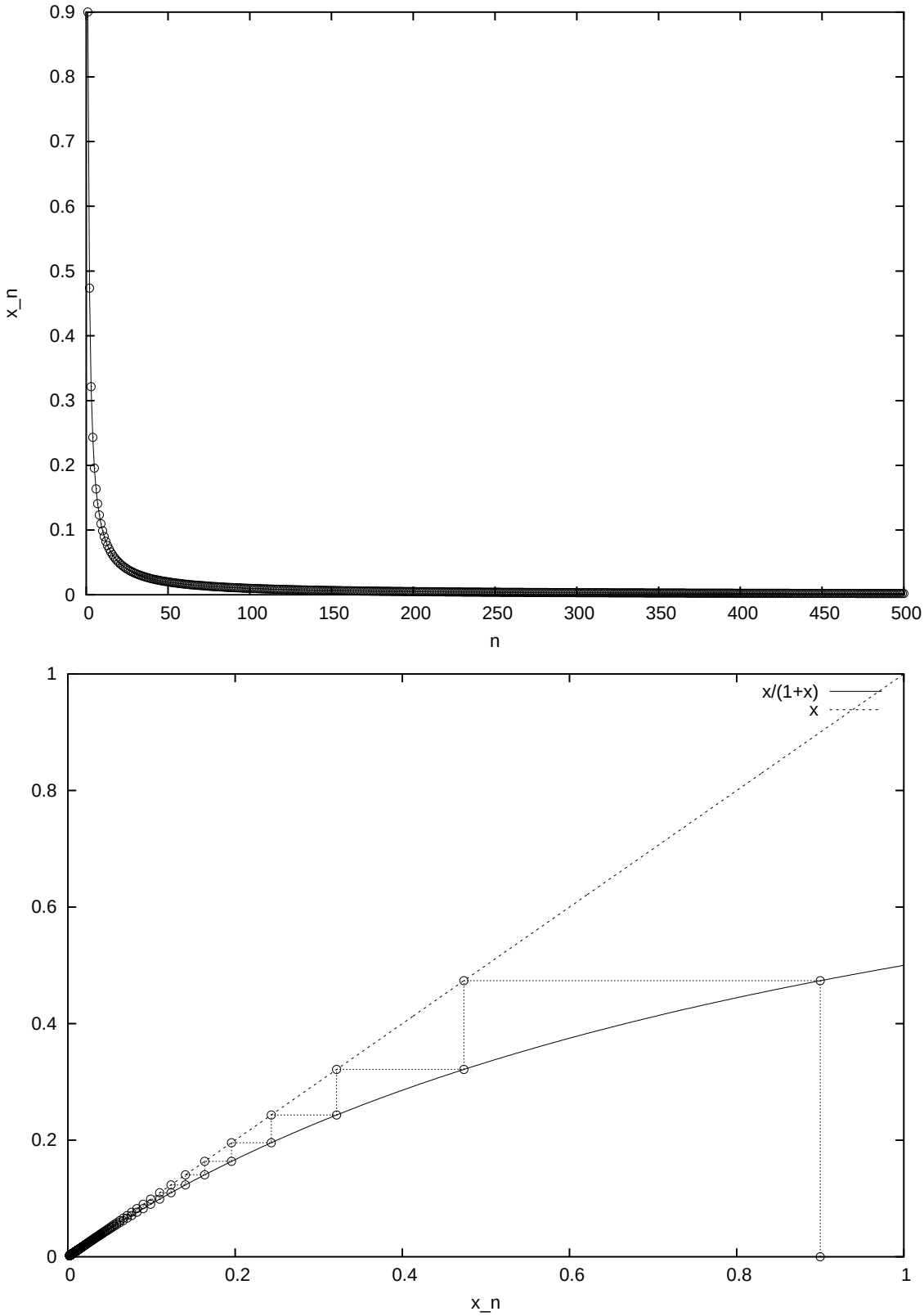


Figura 8: Storia temporale x_n e studio dei punti uniti, $N = 500$, $x_0 = 0.9$.

1.6 Esercizio

Sia $\lambda \in \mathbb{R}$. Si studi la successione definita da

$$x_0 = \lambda, \qquad x_{k+1} = 4\int_0^{x_k} \frac{e^{2\tau}}{(e^{2\tau}+1)^2} d\tau.$$

1.6.1 Risoluzione

Si osservi che $x_{k+1} = 4\displaystyle\int_0^{x_k} \frac{e^{2\tau}}{(e^{2\tau}+1)^2} = 1 - \frac{2}{e^{2x_k}+1}$, pertanto $f(x) = 1 - \dfrac{2}{e^{2x}+1}$. Essa è una funzione sempre crescente che ha come unico punto unito $x = 0$, dove la sua derivata prima $f'(x) = \frac{4\,e^{2\,x}}{(e^{2\,x}+1)^2}$ vale $f'(0) = 1$. Pertanto è necessario l'uso delle derivate successive. Essendo $f''(x) = -\frac{8\,(e^x-1)\,(e^x+1)\,e^{2\,x}}{(e^{2\,x}+1)^3}$ e quindi $f''(0) = 0$ e $f'''(x) = \frac{16\,e^{2\,x}\left(e^{4\,x}-4\,e^{2\,x}+1\right)}{(e^{2\,x}+1)^4}$ e quindi $f'''(0) = -2 < 0$, l'origine è un punto localmente asintoticamente stabile.

Alternativamente, si osservi che $f'(x) > 0\ \forall x \in \mathbb{R}$ e quindi la funzione è sempre crescente. Pertanto anche la successione è monotona. Partendo da $x_0 > 0$ si ha $x_1 < x_0$ e quindi la successione è decrescente e converge a 0. Partendo da $x_0 < 0$ si ha $x_1 > x_0$ e quindi la successione è crescente e converge a 0. Partendo da $x_0 = 0$, x_n rimane tale.

`esempio6.m` può essere utilizzato per la soluzione grafica di questo esercizio. Esso richiede come input N (numero di iterazioni) e il valore iniziale $x_0 = \lambda$.

In figura 9 è riportato un esempio per $N = 500$ e $x_0 = 1.8$.

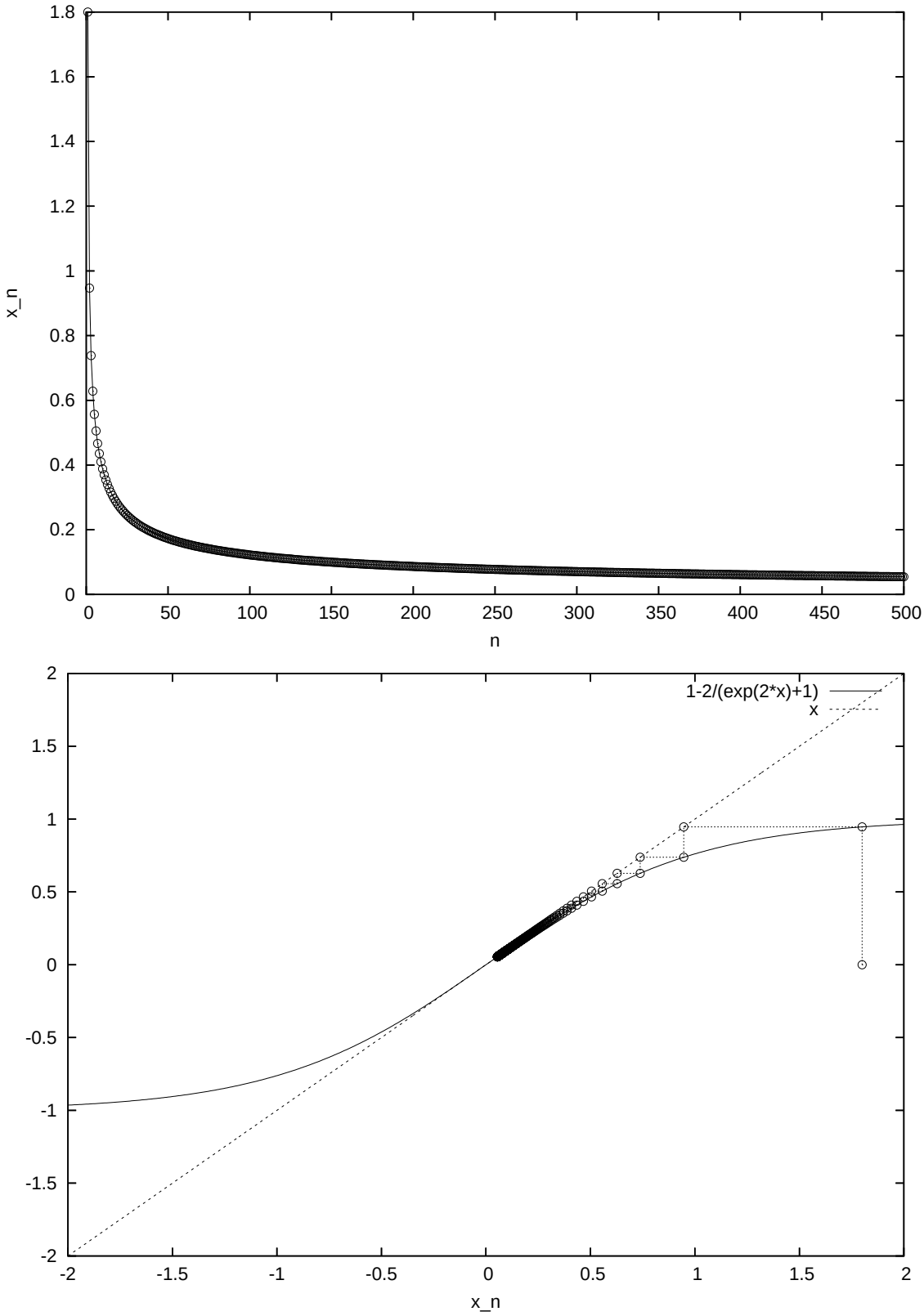


Figura 9: Storia temporale x_n e studio dei punti uniti, $N = 500$ e $x_0 = 1.8$.

1.7 Esercizio

Sia $\lambda > 0$. Si studi la successione definita da

$$x_0 = 0, \qquad x_1 = \lambda, \qquad x_{k+1} = x_k + x_{k-1}^2.$$

1.7.1 Risoluzione

Si noti che la successione è strettamente crescente essendo $x_{k+1} - x_k = x_{k-1}^2 > 0 \; \forall x_{k-1} \neq 0$ e quindi la successione diverge a $+\infty \; \forall x_1 > 0$.

`esempio7.m` può essere utilizzato per la soluzione grafica di questo esercizio. Esso richiede come input N (numero di iterazioni) e il valore iniziale $x_1 = \lambda > 0$.

1.8 Esercizio

Sia $\lambda > 0$. Si studi la successione definita da

$$x_0 = \lambda, \qquad x_{k+1} = \log(1 + x_k).$$

1.8.1 Risoluzione

Si noti che $f(x) = \log(1+x)$ è strettamente crescente pertanto anche la successione (x_k) è monotona. Essendo $x = 0$ l'unico punto unito, e $x_1 < x_0 \; \forall x_0 > 0$, la successione è decrescente e converge a zero. Alternativamente, si osservi che $f'(0) = 1/(1+x)$, quindi $\lim_{x\to 0^\pm} f'(x) = \lim_{x\to 0^\pm} 1/(x+1) = 1^\mp$ e pertanto l'origine è stabile superiormente e instabile inferiormente. Ricorrendo allo schema 5, si ha $f'(0) = 1$, $f''(x) = -1/(1+x)^2 \Rightarrow f''(0) = -1 < 0$, ovvero l'origine è superiormente localmente asintoticamente stabile ed inferiormente repulsiva. Essendo interessati solo al caso $\lambda > 0$, l'origine risulta stabile come riportato in figura 10 per $N = 500$ e $x_0 = 1.8$.

`esempio8.m` può essere utilizzato per la soluzione grafica di questo esercizio. Esso richiede come input N (numero di iterazioni) e il valore iniziale $x_0 = \lambda > 0$.

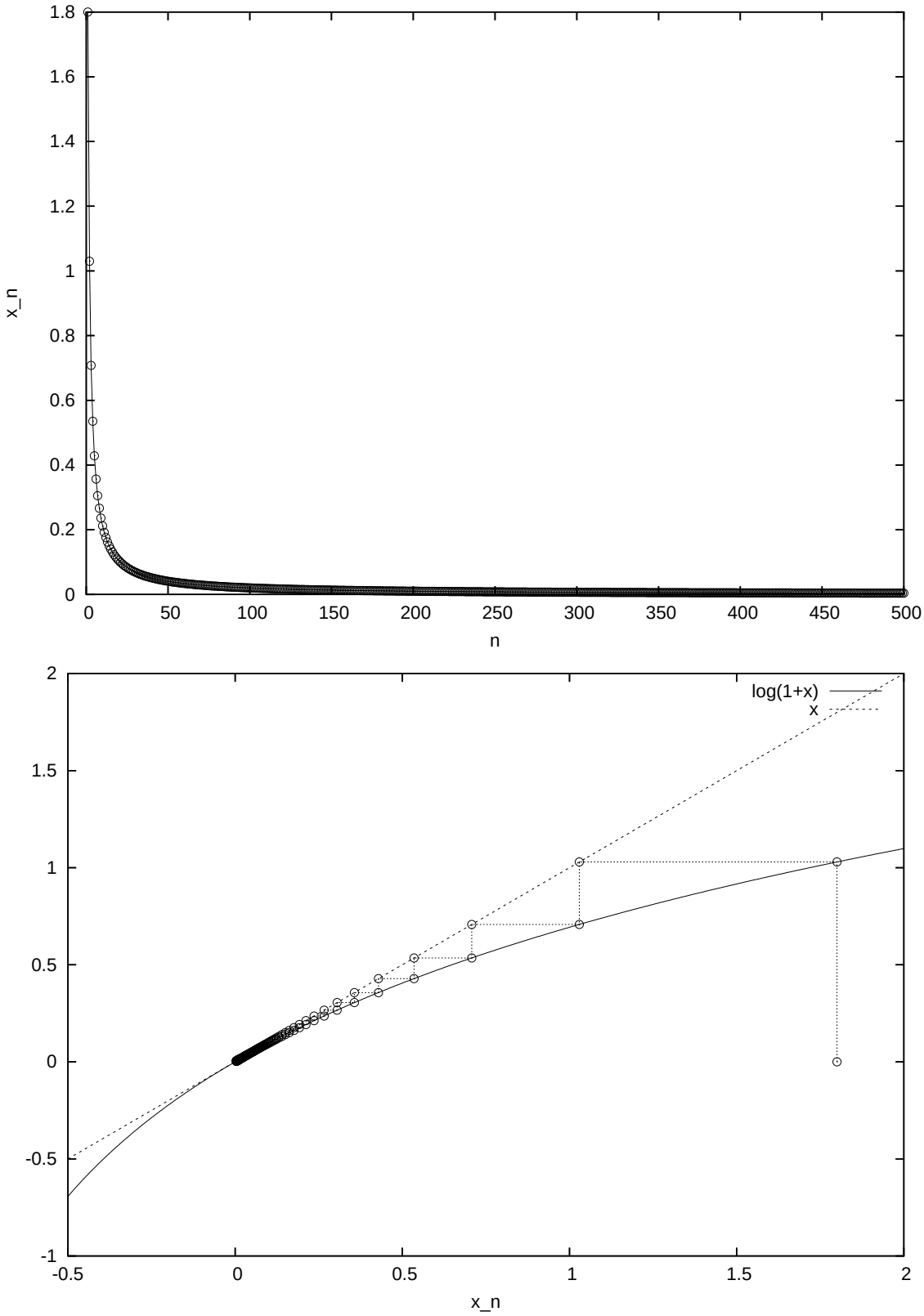


Figura 10: Storia temporale x_n e studio dei punti uniti, $N = 500$ e $x_0 = 1.8$.

2 Simulazione al calcolatore del modello logistico discreto $x_{k+1} = Ax_k(1 - x_k)$

Il modello logistico discreto descrive la legge di crescita di una popolazione "riscalata" x_k al tempo k-esimo secondo la mappa

$$x_{k+1} = Ax_k(1 - x_k). \tag{1}$$

Si noti che, essendo la popolazione normalizzata, affinché il tutto abbia senso, deve essere $0 \leq x_k \leq 1 \; \forall k \geq 0$.

2.1 Esercizio

Determinare i valori di $A \in \mathbb{R}$ affinché la (1) abbia un effettivo significato biologico.

2.1.1 Risoluzione

Siccome popolazioni negative non esistono (ovvero non esistono popolazioni in cui il numero dei morti supera quello dei vivi essendo esse estinte al momento in cui $x_k = 0$), deve essere $A > 0$. Siccome la legge di crescita è $x_{k+1} = f(x_k)$ con $f(x) = Ax(1 - x)$, ovvero una parabola rivolta verso il basso con vertice in $V(1/2; A/4)$, il massimo valore che può assumere x_{k+1} è $A/4$, da cui la limitazione $A/4 \leq 1 \Rightarrow A \leq 4$. Pertanto, $0 < A \leq 4$.

In alternativa, si può verificare facilmente che per $A > 4$ prima o poi si raggiungono valori negativi di x_k indipendentemente da dove si parte con x_0.

2.2 Esercizio

Determinare i punti di equilibrio di (1) e discuterne la stabilità.

2.2.1 Risoluzione

Si tratta di determinare i punti uniti della successione definita per ricorrenza $x_{k+1} = f(x_k)$ con $f(x) = Ax(1 - x)$ e $0 \leq x_0 \leq 1$. Risolvendo $x = f(x)$ si ottengono le soluzioni $x = 0 \; \vee \; x = (1 - 1/A) \; \vee \; x = +\infty$. Chiaramente, $x = +\infty$ non è accettabile. Inoltre, la soluzione $x = 1 - 1/A$ è accettabile, ovvero positiva, solo per $1 - 1/A > 0 \Rightarrow A > 1$. Pertanto, per $0 \leq A \leq 1$ si ha una sola soluzione di equilibrio ($x = 0$) mentre per $1 < A \leq 4$ si hanno 2 punti di equilibrio ($x = 0 \; \vee \; x = 1 - 1/A$). In figura 11 sono riportate le curve $f(x) = Ax(1 - x)$ per alcuni valori significativi del parametro A. Si noti la presenza di una o due soluzioni dipendentemente da A.

Per quanto riguarda la stabilità dei punti di equilibrio, si osservi che $f'(x) = A - 2Ax$. Pertanto $f'(0) = A$, quindi si ha stabilità della soluzione $x = 0$ solo per $0 < A < 1$. Si noti che in tal caso $x = 0$ è l'*unica* soluzione possibile. Pertanto, se $0 < A < 1$ la popolazione è condannata all'estinzione. Nel caso $1 < A \leq 4$, invece, le soluzioni sono 2 e $x = 0$ è instabile. Questo è un fatto positivo in quanto implica che la popolazione non è destinata all'estinzione. Per quanto riguarda la stabilità dell'altro punto di equilibrio

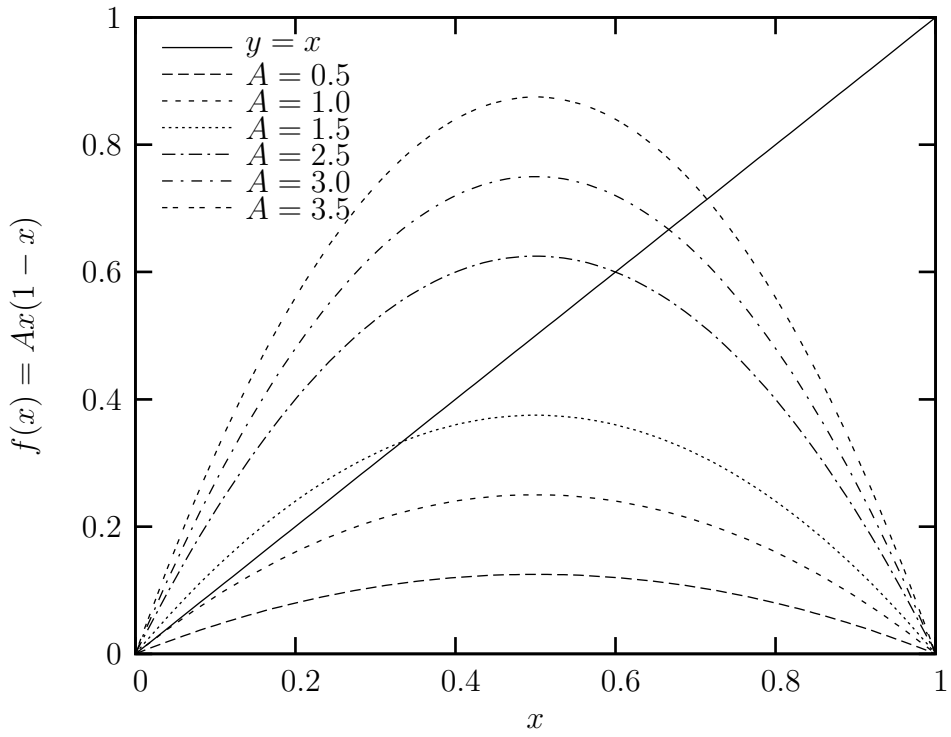


Figura 11: Grafico di $f(x) = Ax(1-x)$ per diversi valori di A.

$x = 1 - 1/A$ (che esiste solo se $1 < A \leq 4$), si ha $f'(1-1/A) = 2 - A$, per cui la condizione di stabilità si traduce in $|2 - A| < 1 \Rightarrow 1 < A < 3$. Pertanto, la soluzione $x = 1 - 1/A$ è stabile per $1 < A < 3$, instabile per $3 \leq A < 4$. Le caratteristiche di stabilità dei punti uniti sono facilmente verificabili guardando la figura 11 e notando che $|f'(1-1/A)| < 1$ per $1 < A < 3$, mentre $|f'(1-1/A)| > 1$ per $3 < A \leq 4$.

Tutte le caratteristiche di (x_k) sopra menzionare sono verificabili graficamente utilizzando il file `logistic.m` riportato in appendice A. Esso richiede come input A (costante positiva), N (numero di iterazioni – quindi nel caso $k \to +\infty$ si deve scegliere N opportunamente elevato) e il valore iniziale x_0 (chiamato `s(1)` in `logistic.m`). Si noti che introducendo $A < 0$ ne viene preso il valore assoluto e che deve essere $0 \leq x_0 \leq 1$ altrimenti viene generato un numero casuale tra 0 e 1.

Si noti che `logistic.m` stampa gli ultimi 8 valori della successione (x_k) al fine di verificare eventuali periodicità.

2.3 Esercizio

Si analizzi numericamente cosa succede per $A = 3 + \epsilon$ e si verifichi che per ϵ abbastanza piccolo si ottengono soluzioni periodiche.

2.3.1 Risoluzione

Si facciano alcune prove con `logistic.m` scegliendo opportunamente i parametri.

2.4 Esercizio

Studiare il comportamento di (1) per $3 < A \leq 4$.

2.4.1 Risoluzione

Come visto in precedenza, in questo intervallo di valori la soluzione $x = 1 - 1/A$ è instabile. Tuttavia, esistono soluzioni stabili per la mappa $x_{k+1} = f^2(x_k)$, dove $f^2(x) = f(f(x)) = A^2x(1-x)(Ax^2 - Ax + 1) = -A^3x^4 + 2A^3x^3 - A^3x^2 - A^2x^2 + A^2x$ è l'iterata seconda della funzione $f(x) = Ax(1-x)$ (lo studente diligente verifichi i passaggi analitici). Dire che l'iterata seconda ha punti di equilibrio stabili equivale a dire che se x_- e x_+ sono le due soluzioni, allora partendo da $x_0 = x_+$ si avrà $x_{2k} = f(x_+)\ \forall k \in \mathbb{N}$ e $x_{2k+1} = x_1 = f(x_+)\ \forall k \in \mathbb{N}$.

Si noti che la mappa $x_{k+1} = f^2(x_k)$ ha significato fintanto che $x_k \in [0,1]\ \forall k \in \mathbb{N}$, pertanto è necessario richiedere $0 \leq f^2(x) \leq 1$. Siccome $f^2(x)$ è nulla in $x = 0$ e $x = 1$, i massimi possono essere identificati con i comuni criteri dell'analisi (studio del segno della derivata prima). In questo modo, si ottengono i due massimi in $x = \dfrac{A \pm \sqrt{A^2 - 2A}}{2A}$, dove la funzione iterata seconda $f^2(x)$ vale $\max[f^2(x)] = A/4$. Dovendo essere $0 \leq \max[f^2(x)] \leq 1$, si trova $0 < A \leq 4$, che è soddisfatta in quanto $3 < A \leq 4$.

I punti uniti di $f^2(x)$ sono $x = 0$, $x = 1 - 1/A$, $x_+ = \dfrac{A + 1 + \sqrt{A^2 - 2A - 3}}{2A}$ e $x_- = \dfrac{A + 1 - \sqrt{A^2 - 2A - 3}}{2A}$. Gli ultimi due esistono solo per $A \geq 3$ e, in tal caso, è comunque assicurato che tutti i punti uniti si trovino nell'intervallo $[0,1]$ (si verifichi per esercizio). In figura 12 sono riportate $f^2(x)$ e $f(x)$ al variare di A. Si noti che il punto unito compreso tra x_+ e x_- è il punto unito $x = 1 - 1/A$ di $f(x)$.

Ai fini della stabilità occorre conoscere la derivata prima di $f^2(x)$ che vale $d[f^2(x)]/dx = g(x) = -4A^3x^3 + 6A^3x^2 - 2A^3x - 2A^2x + A^2$. Essendo $g(0) = A^2$, il punto unito $x = 0$ è instabile per $A > 1$ (lo si sapeva già). In $x = 1 - 1/A$ si ottiene $g(1 - 1/A) = (A-2)^2$, per cui questo punto unito è stabile per $|A - 2| < 1$, ovvero $1 < A < 3$ (si ricordi che per $0 < A < 1$ questo punto unito non esiste), conclusione già nota dall'analisi precedente. Per i nuovi punti uniti, si ottiene $g(x_+) = g(x_-) = -(A^2 - 2A - 4)$, ovvero essi hanno la stessa tangente, e affinchè siano stabili deve essere $|-(A^2 - 2A - 4)| < 1 \Rightarrow 3 < A < 1 + \sqrt{6} \approx 3.45$. Le caratteristiche di stabilità dei punti uniti sono facilmente verificabili guardando la figura 12 e notando che $|f'(x_+) = f'(x_-)| < 1$ per $3 < A < 1 + \sqrt{6}$, mentre $|f'(x_+) = f'(x_-)| > 1$ per $1 + \sqrt{6} < A \leq 4$.

Il comportamento per $1 + \sqrt{6} < A \leq 4$, chiaramente, non può più essere descritto utilizzando l'iterata seconda, ma si deve ricorrere all'iterata terza (fintanto che essa ammette punti di equilibrio stabili), quindi all'iterata quarta e così via. Per $3.45 < A < 3.54$ (si ricordi che questi valori sono approssimativi), x_k oscilla tra 4 valori, mentre per A leggermente superiore a 3.54 la soluzione oscilla tra 8 valori stabili, quindi 16, 32 etc. Questo fenomeno passa sotto il nome di *period-doubling cascade*. Si osserva che questi raddoppi del periodo si susseguono sempre più velocemente e per $A \approx 3.57$ si raggiunge una condizione in cui x_k assume tutti valori diversi con l'impossibilità di riconoscere delle oscillazioni periodiche. Inoltre, piccole variazioni iniziali portano a

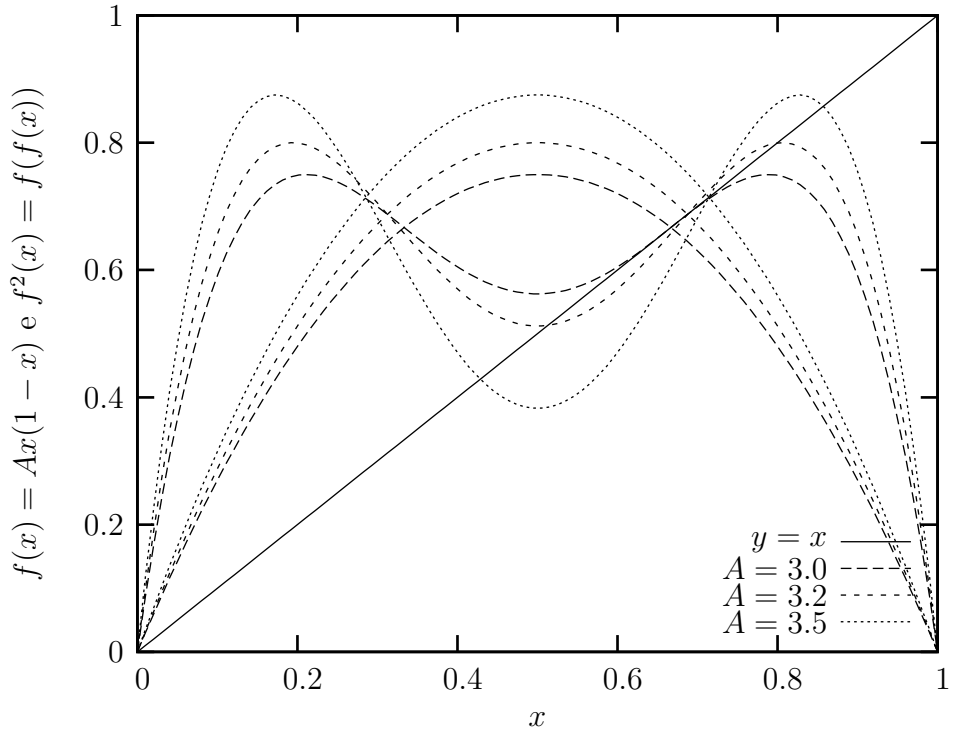


Figura 12: Grafico di $f^2(x) = -A^3x^4 + 2A^3x^3 - A^3x^2 - A^2x^2 + A^2x$ e $f(x) = Ax(1-x)$ per diversi valori di A.

stati finali completamente diversi (sensibilità alle condizioni iniziali). In altre parole, si è raggiunto il caos. In figura 13 è riportato l'andamento dei valori della logistic map $x_{k+1} = Ax_k(1-x_k)$ al variare del parametro A.

Si osservi (figura 13) che per molti valori di $A > 3.57$ il comportamento è caotico, ma per valori isolati di A si nota un comportamento periodico. Queste sono regioni di stabilità delle soluzioni periodiche. Facendo uno zoom in $3.8 < A < 3.9$, vedi figura 14, si nota una di queste regioni intorno ad $A \approx 3.83$, dove dapprima ci sono 3 soluzioni stabili, quindi 6, 12, etc. In altri regioni di stabilità x_k oscilla tra 5 soluzioni.

Le figure 13 e 14 sono state ottenute utilizzando il programma `logisticplot.m` per il quale è necessario anche `population.m`. I listati sono riportati in appendice A.

2.5 Esercizio

Utilizzando `logistic.m`, verificare i seguenti asserti (alcuni dei quali sono stati dimostrati precedentemente in modo analitico).

1. Per $A > 4$ e indipendentemente dal dato iniziale x_0 esiste $k \in \mathbb{N}$ tale che $x_k < 0$.
2. Per $0 < A \leq 1$ il solo punto di equilibrio è $x = 0$ e risulta stabile.
3. Per $1 < A < 3$ esistono due soluzioni di equilibrio, di cui una è $x = 0$ ed è instabile e l'altra è $x = 1 - 1/A$ ed è stabile.

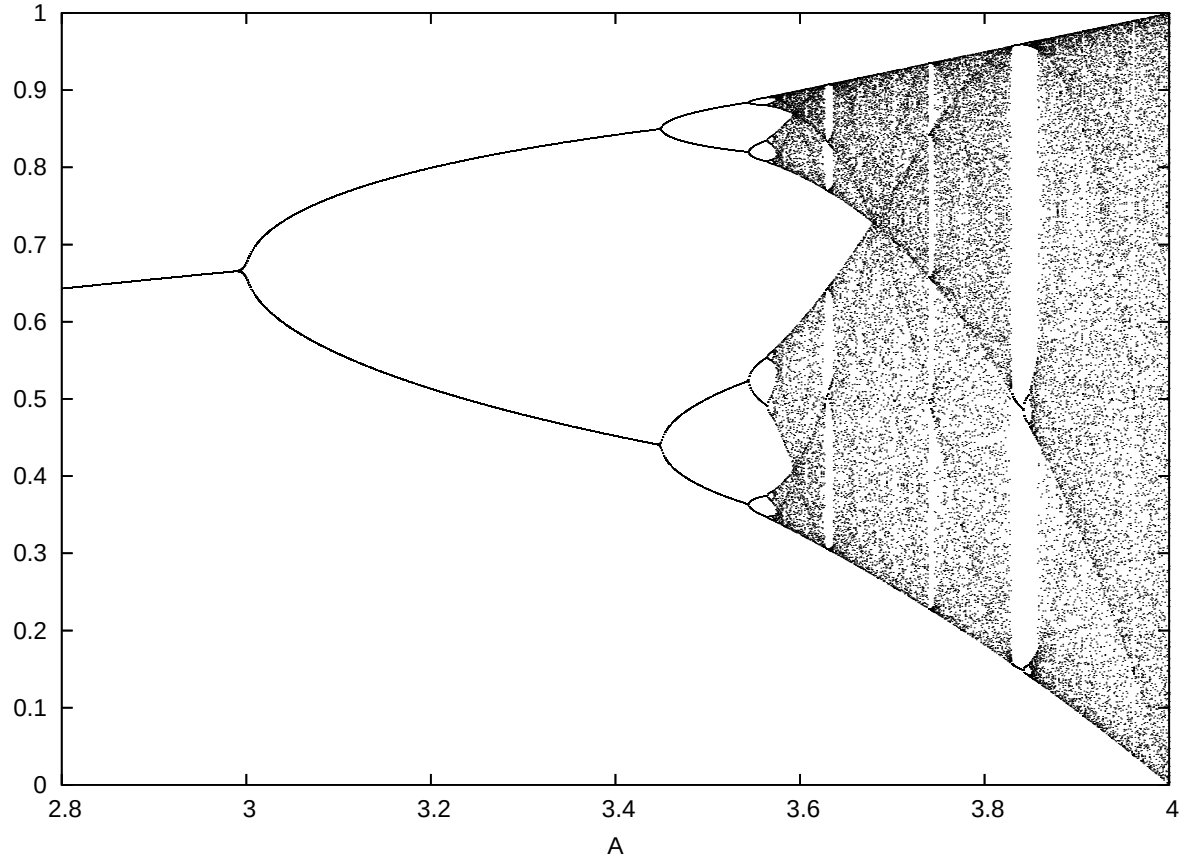


Figura 13: Diagramma delle biforcazioni della logistic map al variare del parametro A.

4. Per $A = 3$ entrambi i punti di equilibrio sono instabili. Cosa succede partendo da $x_0 = 2/3$?

5. Si fissino x_0 ed N (possibilmente molto elevato, 5000). Si discutano i risultati ottenuti per $A \in \{3.39, 3.40, 3.41, 3.42, 3.43, 3.44, 3.45, 3.46\}$.

6. Si trovi il valore di A preciso alla quinta cifra significativa che provoca il secondo raddoppio del periodo.

7. Si fissino $x_0 = 0.1$ ed $N = 1000$ e si discutano i risultati ottenuti per $A \in \{3.6, 4.0\}$.

8. Fissati x_0 ed N, discutere i risultati per $A \approx 3.83$ e $A \approx 3.845$.

2.5.1 Risoluzione

1. Si fissi $A = 4 + \epsilon$ (per esempio $A = 4.001$) e il dato iniziale (per esempio $x_0 = 0.1$) e si facciano diverse prove al variare di N. Se A cresce, $x_k < 0$ viene raggiunto prima o dopo?

2. Fare diverse prove al variare di $A \in]0, 1[$ e $x_0 \in]0, 1[$.

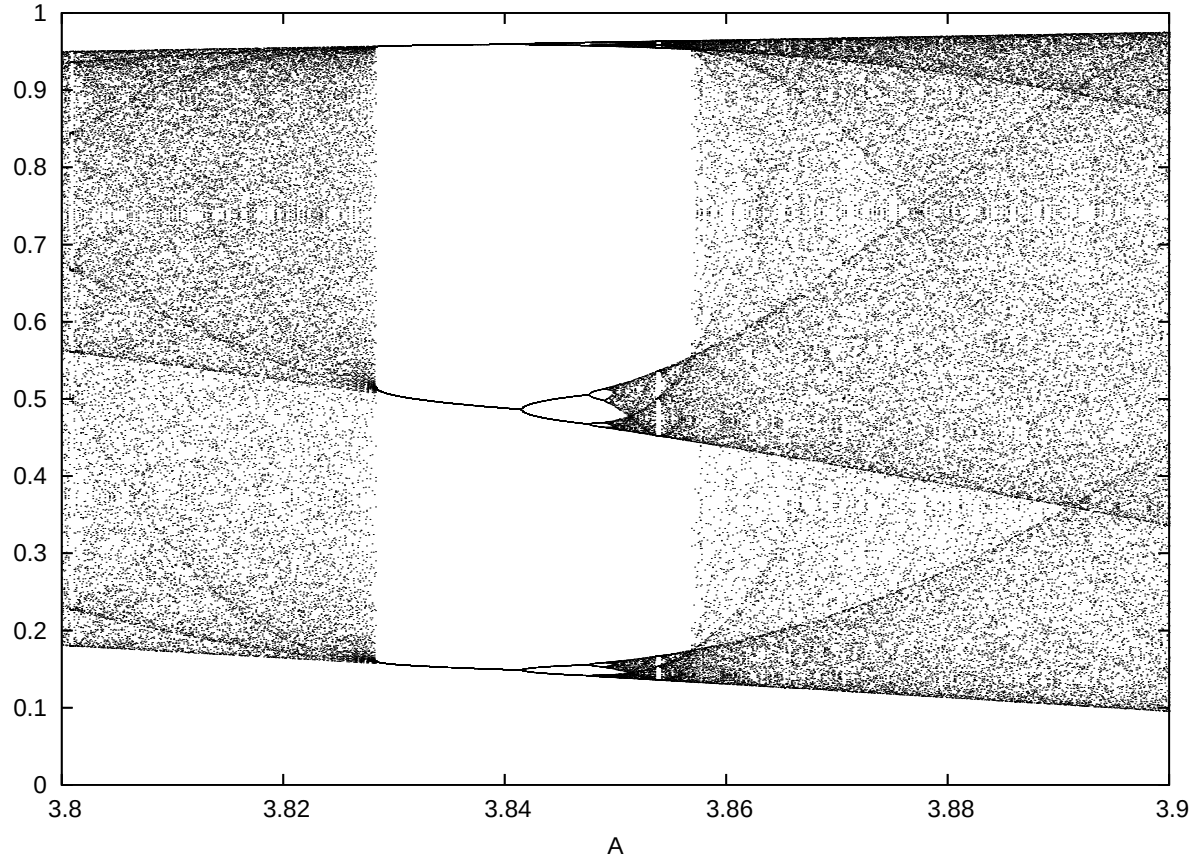


Figura 14: Zoom del diagramma delle biforcazioni della logistic map al variare del parametro A. Si noti la presenza di una regione con soluzioni periodiche per $A \approx 3.83$.

3. Scegliere diversi valori di $A \in]1, 3[$ e $x_0 \in]0, 1[$, verificando che si converge sempre al numero $1 - 1/A$.

4. Facile.

5. Si dovrebbero vedere la successione oscillare prima tra 2 valori e poi tra 4.

6. Per $A = 3.4121$ si notano 4 valori diversi, per $A = 3.4120$ solo 2. Quindi $A = 3.4121$.

7. Per $A = 3.6$ si nota una regione limitata visitata dalla soluzione, per $A = 4.0$ praticamente x_k assume tutti i valori possibili.

8. Si dovrebbero ottenere rispettivamente una soluzione periodica di periodo 3 e una di periodo 6.

3 Ritratti di fase e traiettorie di alcuni sistemi dinamici simulati al calcolatore

3.1 Esercizio

Studiare, al variare delle condizioni iniziali e del tempo finale, il comportamento del sistema

$$\begin{cases} x' &= y + x(x^2 + y^2 - 1) \\ y' &= -x + y(x^2 + y^2 - 1) \end{cases}$$

3.1.1 Risoluzione

Si verifica facilmente che l'unico punto di equilibrio del sistema è l'origine. Procedendo con l'analisi agli autovalori del problema linearizzato si ottiene lo Jacobiano

$$J(x, y) = \begin{pmatrix} y^2 + 3x^2 - 1 & 2xy + 1 \\ 2xy - 1 & 3y^2 + x^2 - 1 \end{pmatrix},$$

che calcolato nell'origine risulta

$$J(0, 0) = \begin{pmatrix} -1 & 1 \\ -1 & -1 \end{pmatrix}$$

ed ammette i due autovalori complessi $\lambda_{1,2} = -1 \pm i$, entrambi con parte reale negativa. Pertanto, l'origine è un fuoco stabile (si noti che queste conclusioni hanno valore locale). Inoltre, i punti della circonferenza con centro nell'origine e raggio unitario formano un ciclo instabile. Per verificarlo, basta moltiplicare la prima equazione per x, la seconda per y, e sommare membro a membro le due equazioni ottenute. Indicando con $\rho = \sqrt{x^2 + y^2}$ (i.e. la distanza dall'origine, come per le coordinate polari), si ottiene l'equazione differenziale ordinaria $\rho\,\rho' = \rho^2(\rho^2 - 1)$. Ricordando che $\rho > 0$ se $x \neq 0 \wedge y \neq 0$, la funzione $\rho(t)$ risulta decrescente per $0 < \rho < 1$ e crescente per $\rho > 1$, pertanto le traiettorie collassano nell'origine se il dato iniziale ha distanza da essa minore di 1, mentre esplodono in tempo finito se si parte da dati iniziali con distanza dall'origine maggiore di 1 (fuori dalla circonferenza unitaria). Da queste considerazioni si deduce facilmente che la circonferenza unitaria è un ciclo limite instabile. La soluzione non esiste per tempi grandi e, numericamente, si può facilmente determinare il tempo di *blow up*.

Un modo alternativo, e squisitamente analitico, per capire la natura della circonferenza unitaria è il seguente. Assumendo di prendere un dato iniziale (x, y) perturbato rispetto ad un punto sulla circonferenza si ottiene $x^2 + y^2 - 1 = \epsilon$, dove ϵ è piccolo a piacere ma può essere positivo (se la perturbazione porta il punto fuori dal cerchio unitario) o negativo (se il punto perturbato si trova all'interno). Il sistema diventa quindi

$$\begin{cases} x' &= y + x\epsilon \\ y' &= -x + y\epsilon \end{cases}$$

che ammette l'origine come punto di quilibrio. L'analisi agli autovalori del sistema (che è già lineare) fornisce $\lambda_{1,2} = \epsilon \pm i$. Pertanto l'origine è stabile se $\epsilon < 0$, instabile se $\epsilon > 0$,

che conferma quanto dedotto dall'equazione differenziale ordinaria per $\rho = \sqrt{x^2 + y^2}$. La circonferenza $x^2 + y^2 = 1$ è quindi instabile perchè partendo vicino ad essa ci si allontana al crescere di t.

Di seguito sono riportati alcuni esempi che evidenziano la natura dell'origine come fuoco stabile (figura 15), il ciclo limite instabile $x^2 + y^2 = 1$ (figura 16 – si noti che la soluzione si allontana dalla circonferenza solo a causa degli errori numerici), e la divergenza del sistema in tempo finito (figure 17 e 18). Si noti, confrontando le figure 17 e 18, che il tempo di blow-up diminuisce al crescere della distanza dall'origine per la condizione iniziale.

Utilizzando il file `sys1.m` riportato in appendice A si possono fare diverse prove al variare della condizione iniziale e del tempo finale di integrazione. **Si ricordi di cambiare gli estremi della finestra di visualizzazione dipendentemente dalla condizione iniziale**.

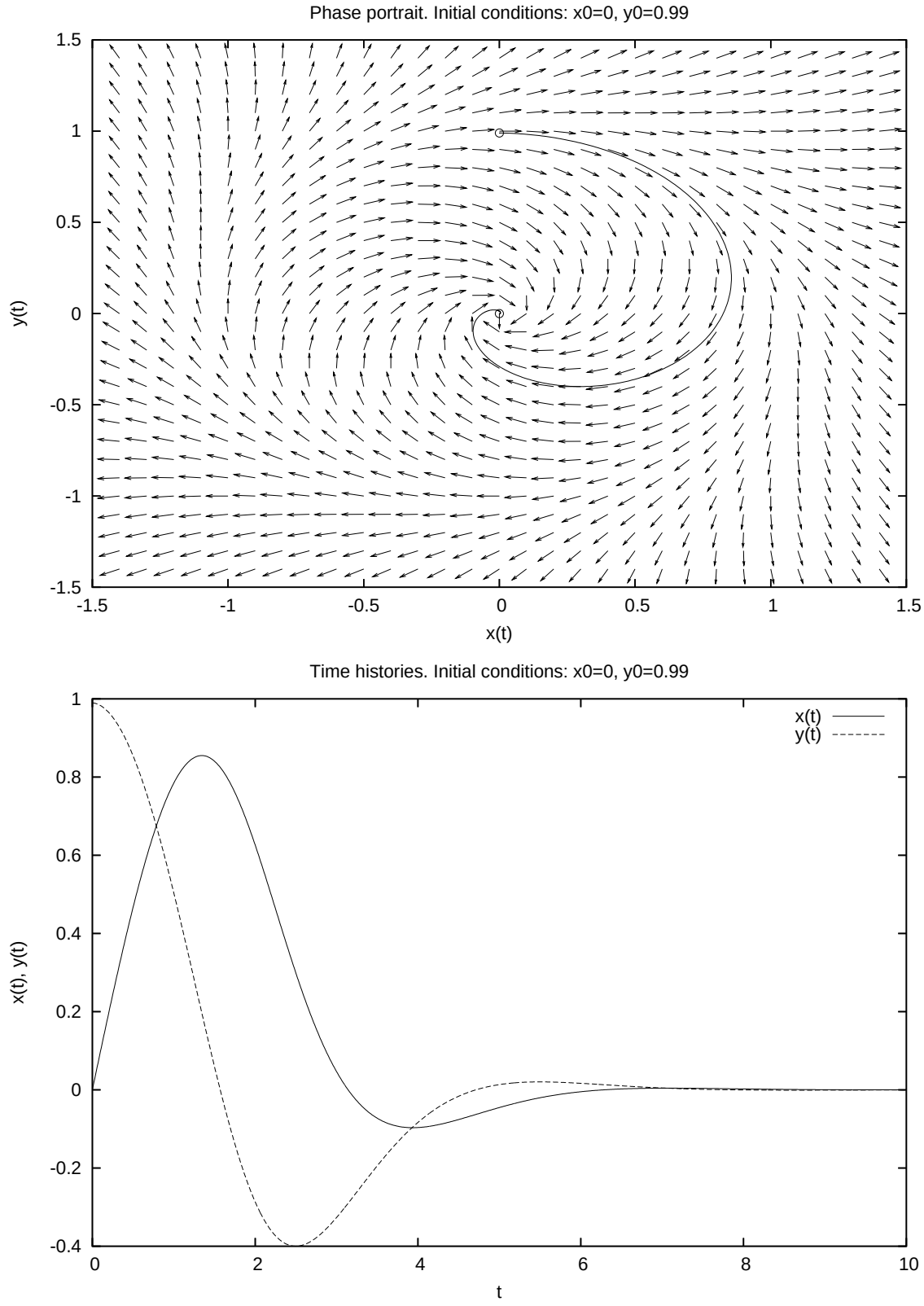


Figura 15: Ritratto di fase e storia temporale per $(x_0, y_0) = (0, 0.99)$ e $t_f = 10$.

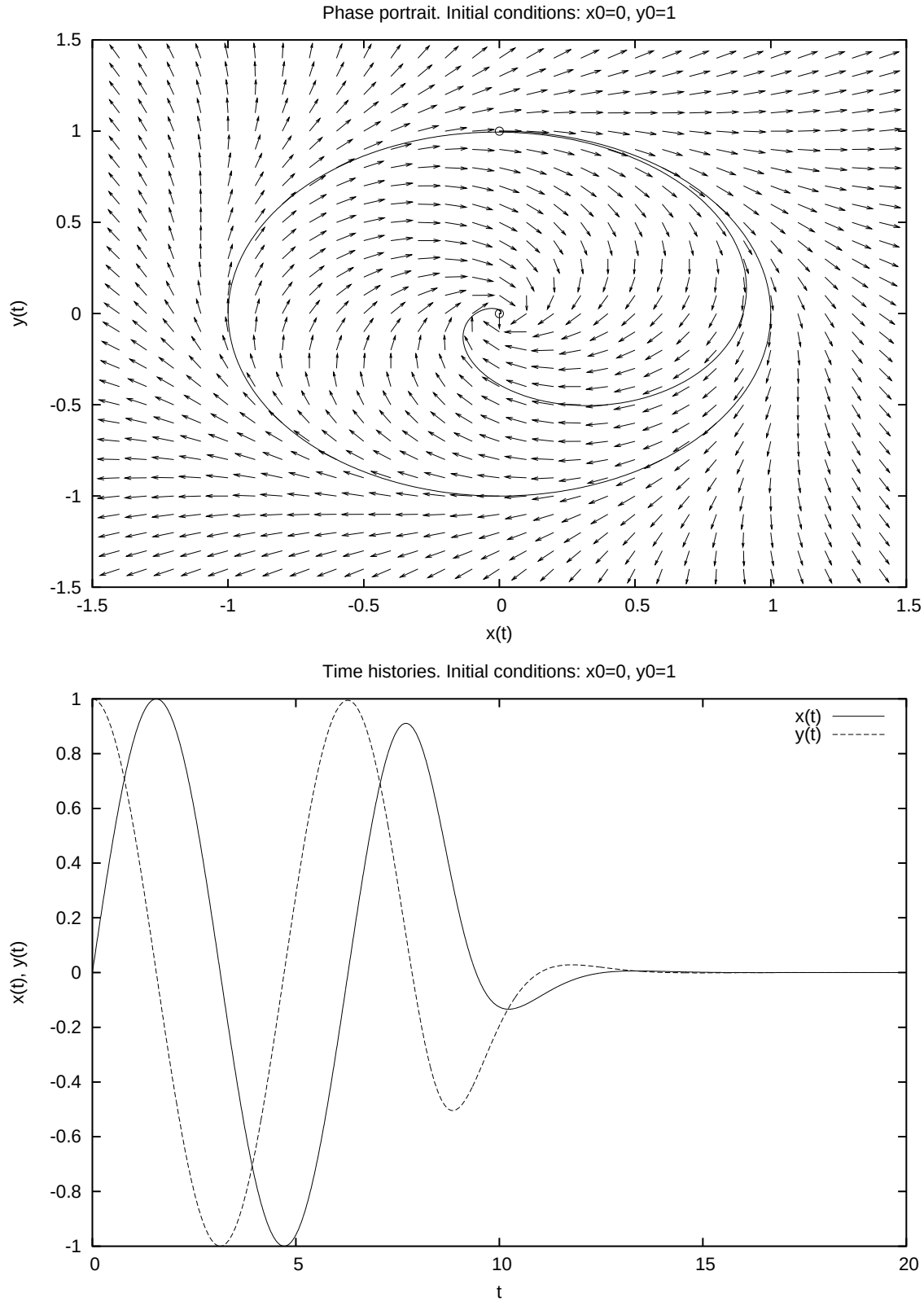


Figura 16: Ritratto di fase e storia temporale per $(x_0, y_0) = (0, 1)$ e $t_f = 20$.

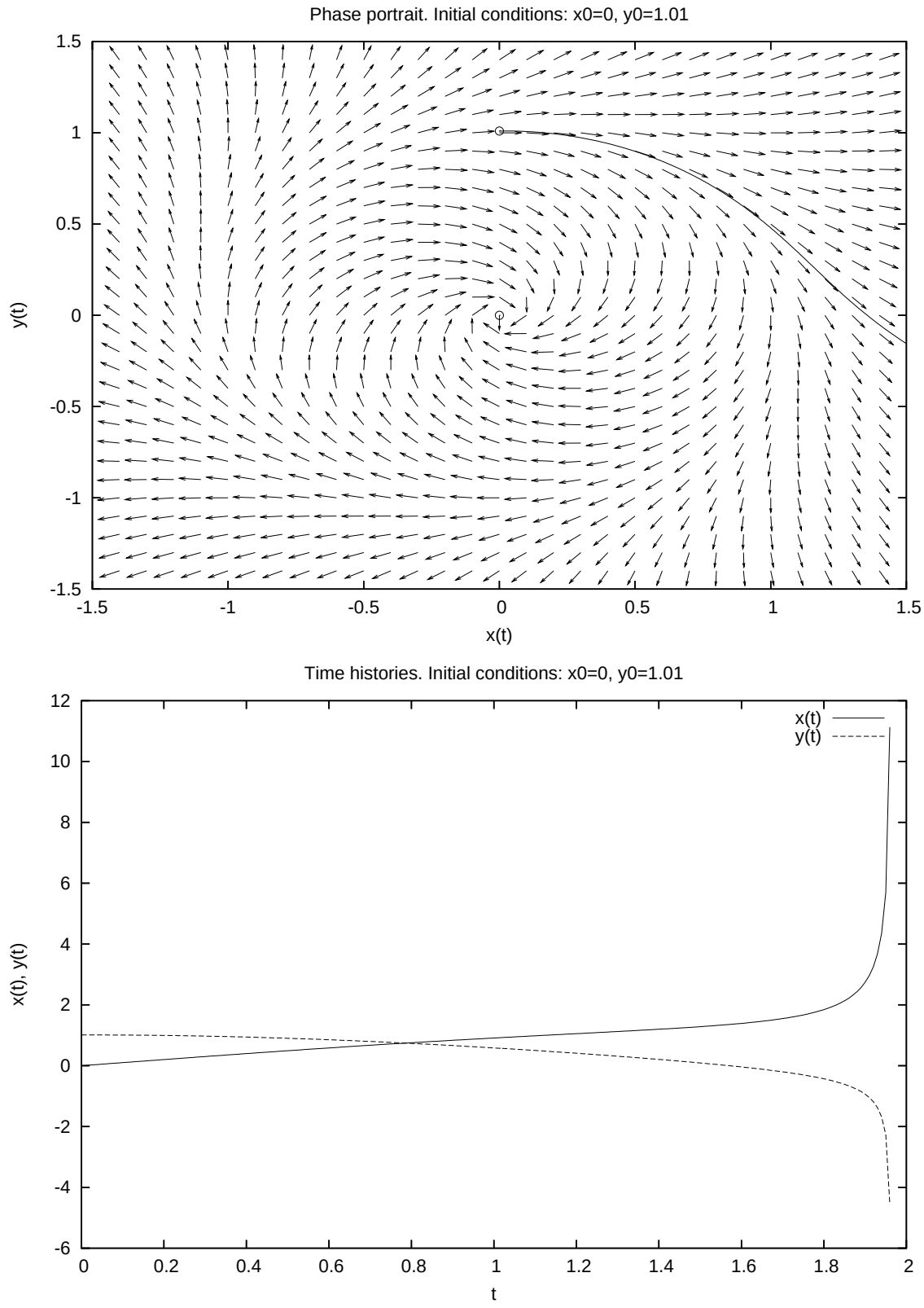


Figura 17: Ritratto di fase e storia temporale per $(x_0, y_0) = (0, 1.01)$ e $t_f = 1.96$.

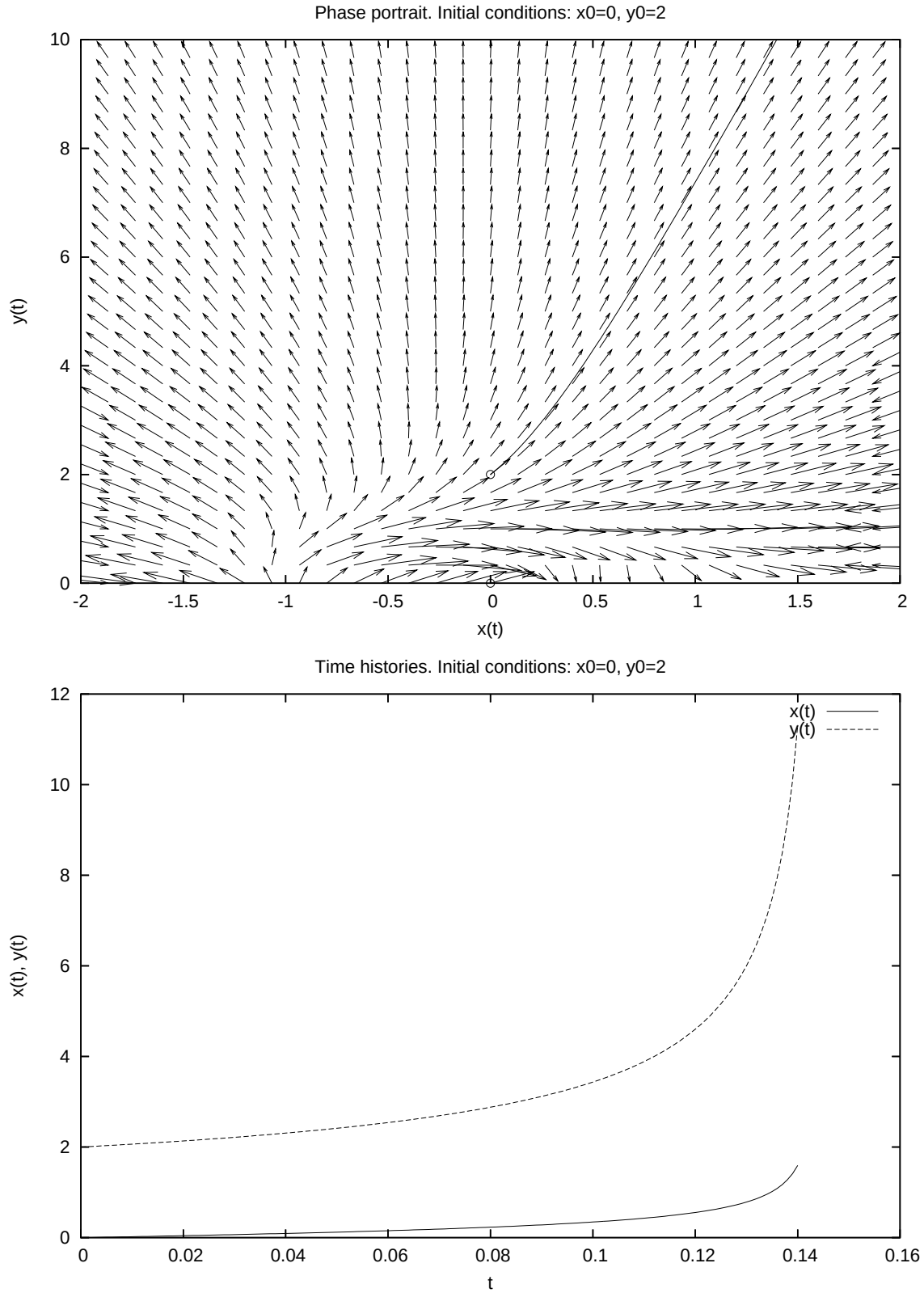


Figura 18: Ritratto di fase e storia temporale per $(x_0, y_0) = (0, 2)$ e $t_f = 0.14$.

3.2 Esercizio

Studiare, al variare delle condizioni iniziali e del tempo finale, il comportamento del sistema

$$\begin{cases} x' &= y + x(\cos(x^2+y^2)-1)(\sin(x^2+y^2)-1) \\ y' &= -x + y(\cos(x^2+y^2)-1)(\sin(x^2+y^2)-1) \end{cases}$$

3.2.1 Risoluzione

L'unico punto di equilibrio per il sistema dato è l'origine. Per capirne la natura, basta seguire l'analisi standard degli autovalori del sistema linearizzato. Così facendo si ottengono entrambi gli autovalori complessi a parte reale positiva. L'origine è quindi un fuoco instabile. Alternativamente, si può studiare l'equazione differenziale ordinaria della distanza dall'origine ($\rho = \sqrt{x^2+y^2}$) ricavata moltiplicando la prima equazione per x, la seconda per y, e sommandole. Così facendo si ottiene $\rho\,\rho' = \rho^2(\cos(\rho^2)-1)(\sin(\rho^2)-1)$, da cui, dopo aver diviso per $\rho > 0$, si ottiene $\rho' > 0$ sempre, tranne quando $\rho^2 = 2k\pi$ oppure $\rho^2 = \pi/2 + k\pi, k \in \mathbb{Z}^+$. Questo significa che la distanza dall'origine non diminuisce mai ma le orbite vengono catturate da circonferenze del tipo

$$x^2+y^2 = 2k\pi \quad \text{e} \quad x^2+y^2 = \pi/2 + k\pi, \qquad k \in \mathbb{Z}^+$$

che sono, quindi, dei cicli semi-stabili.

Utilizzando il file `sys2.m` riportato in appendice A si possono visualizzare i diversi comportamenti al variare della condizione iniziale e del tempo finale di integrazione. In particolare, si può analizzare cosa succede partendo da condizioni iniziali (x_0, y_0) tali per cui la loro distanza dall'origine $\rho_0 = \sqrt{x_0^2+y_0^2}$ varia in certi range. Per fissare le idee, si può prendere $x_0 = 0$ e variare y_0.

- $0 < \rho_0 < 0.25$. Si osserva un allontanamento dall'origine piuttosto lento, come riportato in figura 19.
- $0 < \rho_0 < \sqrt{\pi/2} \approx 1.253$. Si osserva un allontanamento dall'origine piuttosto veloce e lo stabilizzarsi sull'orbita distante dall'origine $\sqrt{\pi/2}$, come riportato in figura 20.
- $\sqrt{\pi/2} \approx 1.253 < \rho_0 < \sqrt{2\pi} \approx 2.5$. Si osserva un allontanamento dall'origine piuttosto veloce e lo stabilizzarsi sull'orbita distante dall'origine $\sqrt{2\pi}$, come riportato in figura 21.

Si ricordi di cambiare gli estremi della finestra di visualizzazione dipendentemente dalla condizione iniziale.

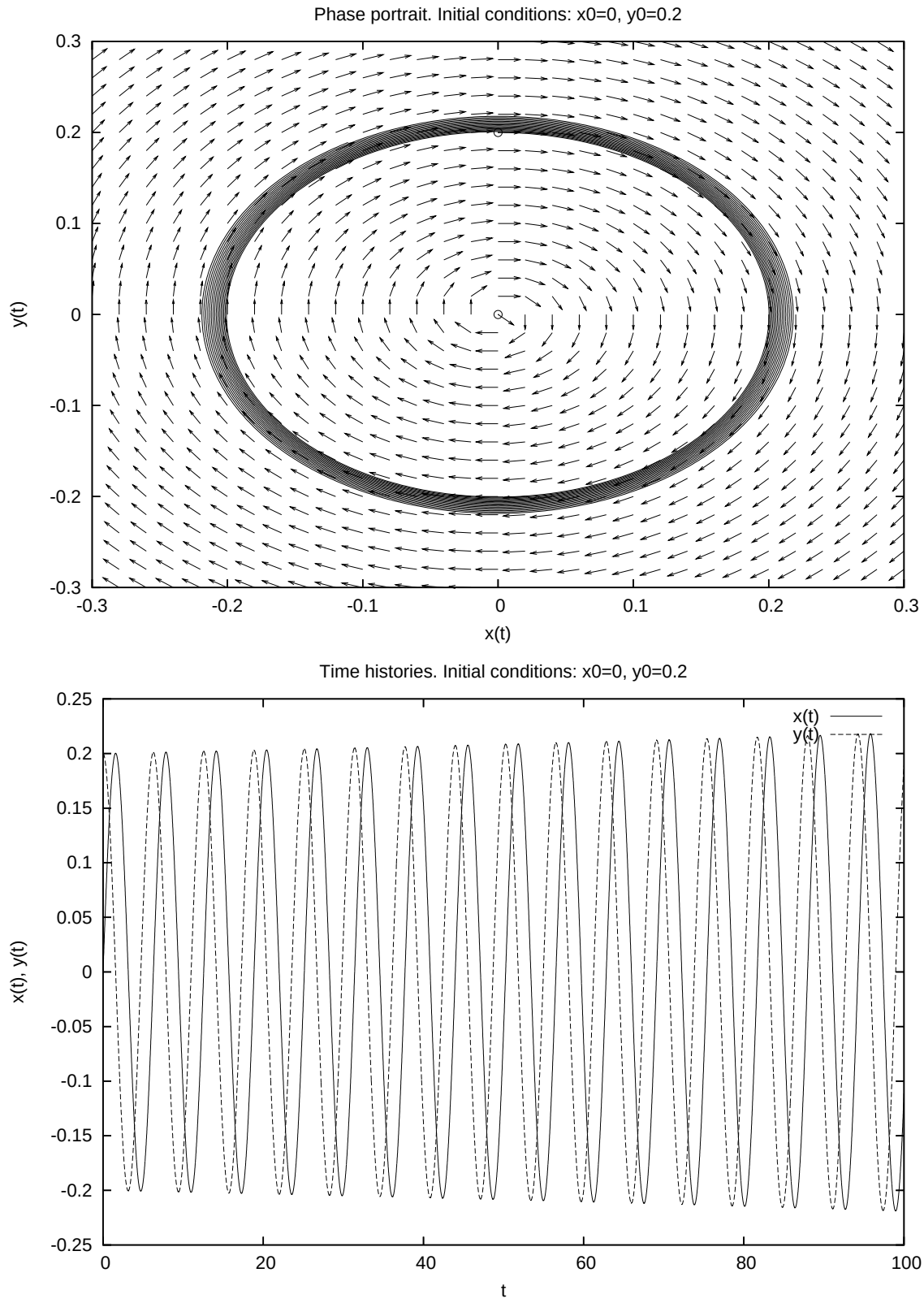


Figura 19: Ritratto di fase e storia temporale per $(x_0, y_0) = (0, 0.2)$ e $t_f = 100$.

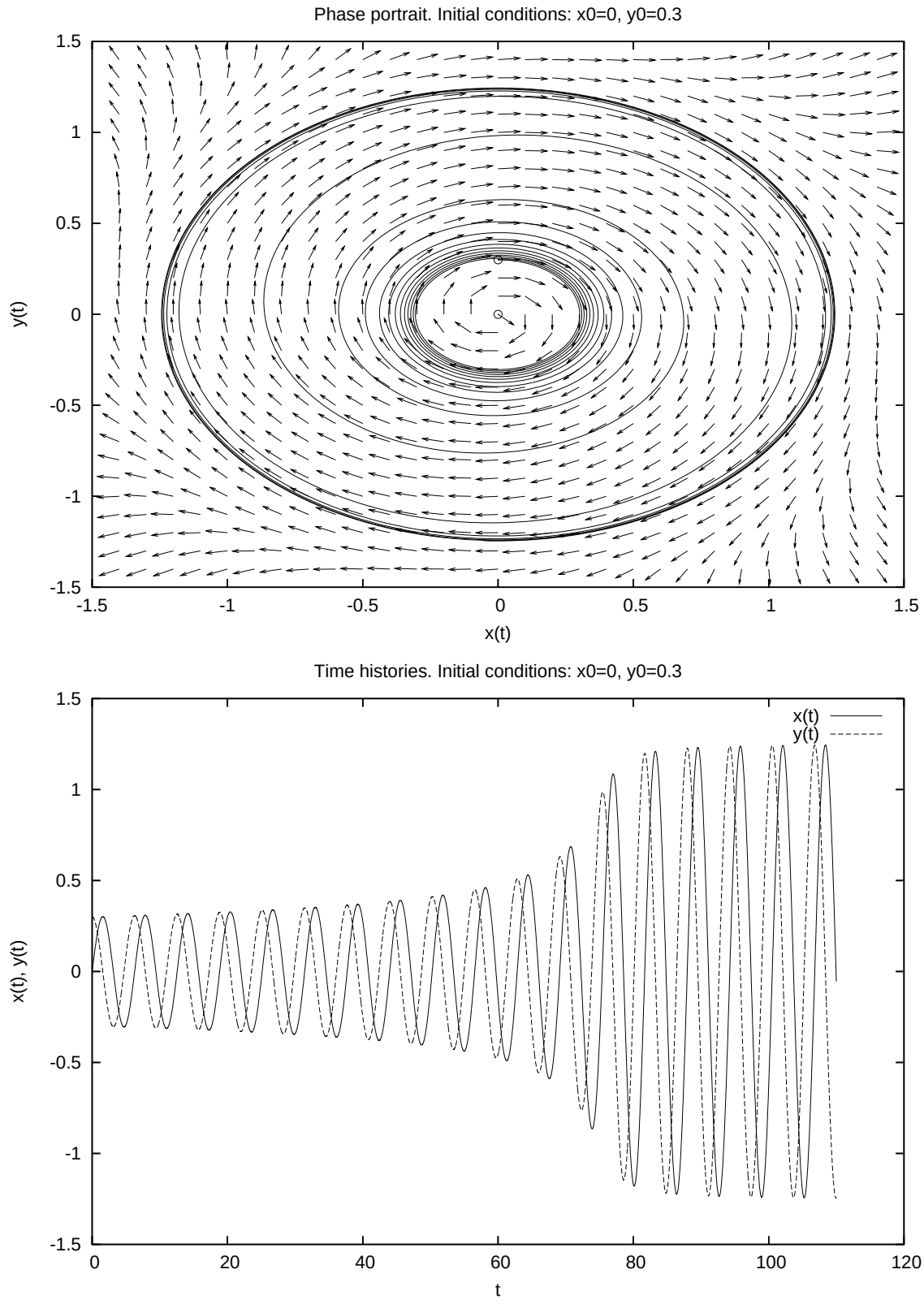


Figura 20: Ritratto di fase e storia temporale per $(x_0, y_0) = (0, 0.3)$ e $t_f = 110$.

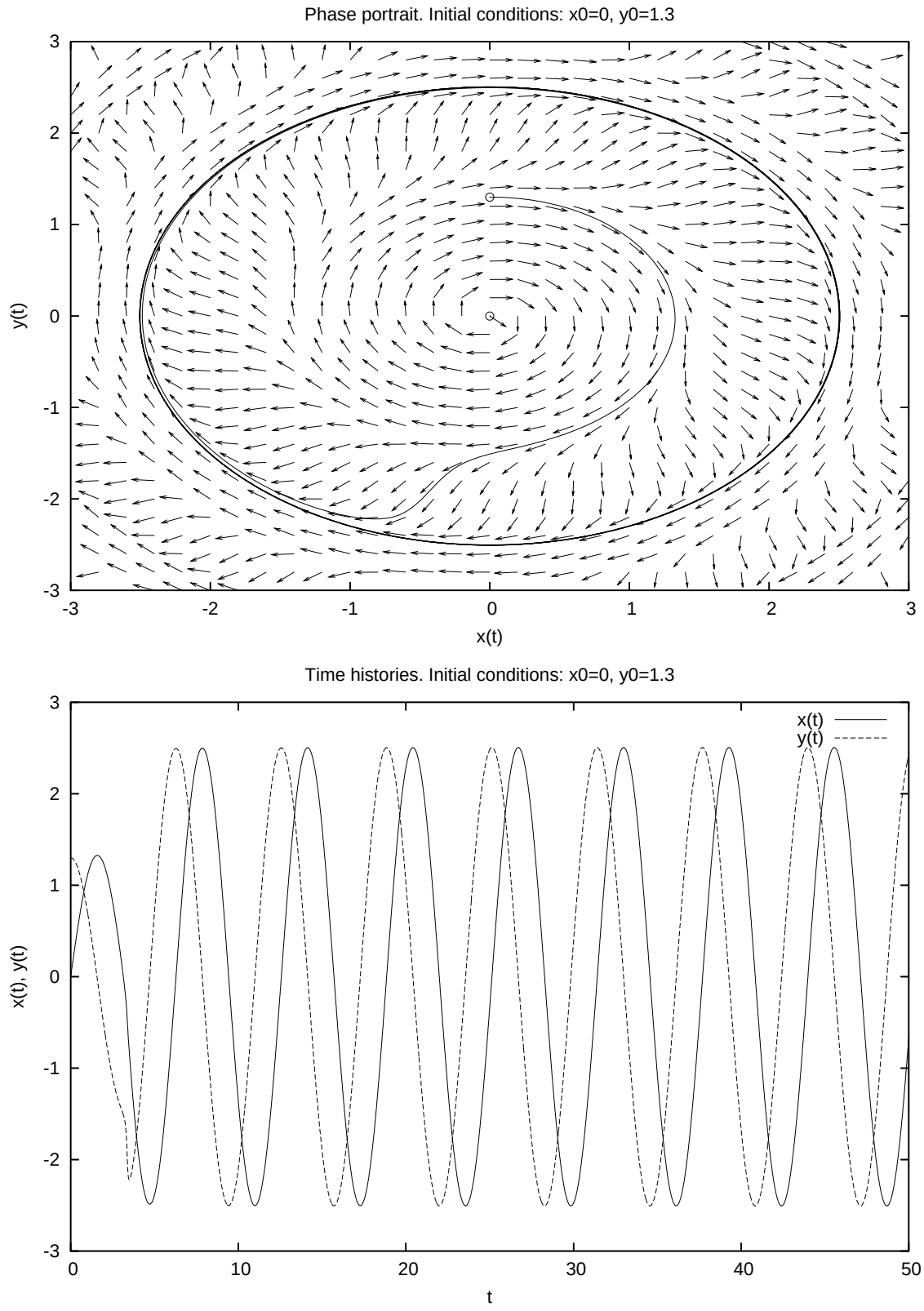


Figura 21: Ritratto di fase e storia temporale per $(x_0, y_0) = (0, 1.3)$ e $t_f = 50$.

3.3 Esercizio

Studiare, al variare delle condizioni iniziali e del tempo finale, il comportamento del sistema

$$\begin{cases} x' &= -y + x(x^2+y^2)\cos(\pi/(x^2+y^2)) \\ y' &= x + y(x^2+y^2)\cos(\pi/(x^2+y^2)) \end{cases}$$

3.3.1 Risoluzione

L'unico punto di equilibrio per il sistema dato è l'origine. Per capirne la natura si può procedere con l'analisi agli autovalori del sistema linearizzato oppure studiare l'equazione differenziale ordinaria della distanza dall'origine ($\rho = \sqrt{x^2+y^2}$) ricavata moltiplicando la prima equazione per x, la seconda per y, e sommandole. Così facendo si ottiene $\rho\,\rho' = \rho^4\cos(\pi/\rho^2)$. Quindi, la distanza dall'origine può aumentare o diminuire dipendentemente dal segno di $\cos(\pi/\rho^2)$. In particolare, le orbite per le quali è costante la distanza dall'origine si ottengono da $\cos(\pi/\rho) = 0 \Rightarrow x^2+y^2 = 2/(2k+1)$, ovvero sono circonferenze concentriche aventi centro nell'origine e raggio $\sqrt{2/(2k+1)}$. La più ampia di queste circonferenze ha raggio $r = \sqrt{2}$, mentre al limite ($k \to +\infty$) il raggio tende a zero.

Per ottenere gli intervalli in cui la triettorie diminuiscono la loro distanza dal centro, basta risolvere la disequazione $\cos(\pi/\rho^2) < 0$, che dopo alcuni passaggi porta a $\sqrt{2/(4k+3)} < \rho < \sqrt{2/(4k+1)}$. Ovviamente, per valori di ρ positivi e complementari ai precedenti le traiettorie si allontanano dall'origine, i.e. per $\sqrt{2/(4k+5)} < \rho < \sqrt{2/(4k+3)}$. Riassumendo:

- per condizioni iniziali aventi distanza dall'origine $\rho_0 > \sqrt{2}$ le traiettorie divergono molto velocemente;
- per condizioni iniziali tali per cui $\sqrt{2/(4k+3)} < \rho_0 < \sqrt{2/(4k+1)}$, ovvero comprese in certe corone circolari, la distanza dall'origine decresce ma vengono catturate dalla circonferenza più interna della corona avente raggio $r = \sqrt{2/(4k+3)}$;
- per condizioni iniziali tali per cui $\sqrt{2/(4k+5)} < \rho_0 < \sqrt{2/(4k+3)}$, ovvero comprese nelle corone circolari complementari alle precedenti, la distanza dall'origine cresce ma vengono catturate dalla circonferenza più esterna della corona avente raggio $r = \sqrt{2/(4k+3)}$.

Utilizzando il file `sys3.m`, riportato in appendice A, si facciano diverse prove al variare della condizione iniziale e del tempo finale di integrazione. In particolare, si discuta cosa succede partendo da condizioni iniziali (x_0, y_0) tali per cui la loro distanza dall'origine $\rho_0 = \sqrt{x_0^2+y_0^2}$ varia in certi range. Per fissare le idee, si può prendere $x_0 = 0$ e variare y_0. **Si ricordi di cambiare gli estremi della finestra di visualizzazione dipendentemente dalla condizione iniziale**.

- $\rho_0 > \sqrt{4}$. Si osserva un rapido allontanamento dall'origine come riportato in figura 22.

- $\sqrt{2/(4k+3)} < \rho_0 < \sqrt{2/(4k+1)}$. Se si prende $k = 0$ questo significa $\sqrt{2/3} < \rho_0 < \sqrt{2}$, e si ottiene un rapido collasso sull'orbita più interna, i.e. la circonferenza avente raggio $\rho_0 = \sqrt{2/(4k+3)} \approx 0.81650$, come mostrato in figura 23.

- $\sqrt{2/(4k+5)} < \rho_0 < \sqrt{2/(4k+3)}$. Se si prende $k = 0$ questo significa $\sqrt{2/5} < \rho_0 < \sqrt{2/3}$, e si ottiene un rapido collasso sull'orbita più esterna, i.e. la circonferenza avente raggio $r\rho_0 = \sqrt{2/(4k+3)} \approx 0.81650$, come mostrato in figura 24.

- Limite per $r \to 0$. In questo caso si parte da condizioni iniziali vicine all'origine, per cui il comportamento è determinato da quanto succede per $k \to +\infty$. Si osserva che, al tendere di k a $+\infty$, le quantità $\sqrt{2/(4k+5)}$, $\sqrt{2/(4k+3)}$, $\sqrt{2/(4k+1)}$ tendono tutte a 0 e si "compattano" molto velocemente. Questo significa che, indipendentemente da dove si parte, si finisce su una circonferenza prossima alla condizione iniziale. In questo senso, l'origine è stabile ma non asintoticamente poiché le triettorie possono allontanarsi o avvicinarsi ad essa (dipendentemente dalla discussione precedente) ma rimangono comunque intrappolate in corone di raggio arbitrariamente piccolo. Per esempio, in figura 25 è riportato il caso della condizione iniziale $(x_0, y_0) = (0, 0.0703)$, che rimane intrappolata nelle corone di raggio $r_1 = 0.0702728368926307$ e $r_2 = 0.0704469953676347$, che corrispondono a $k = 100$ e quindi viene attratta verso la circonferenza più esterna, come si può facilmente notare scrivendo a schermo la distanza dall'origine per l'ultimo punto calcolato con il comando `sqrt(x(1,size(x)(2))^2+x(2,size(x)(2))^2)`.

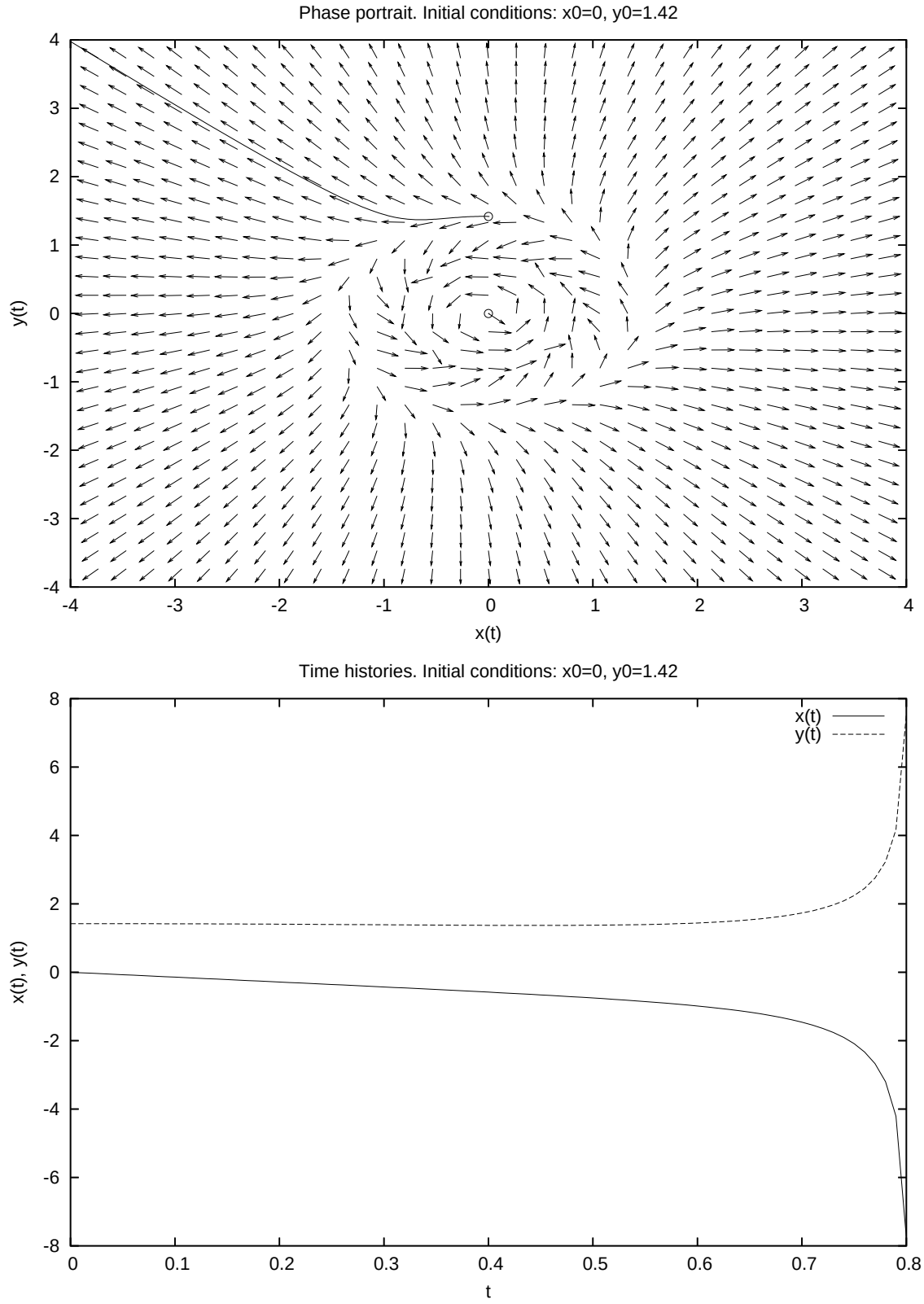


Figura 22: Ritratto di fase e storia temporale per $(x_0, y_0) = (0, 1.42)$ e $t_f = 0.8$.

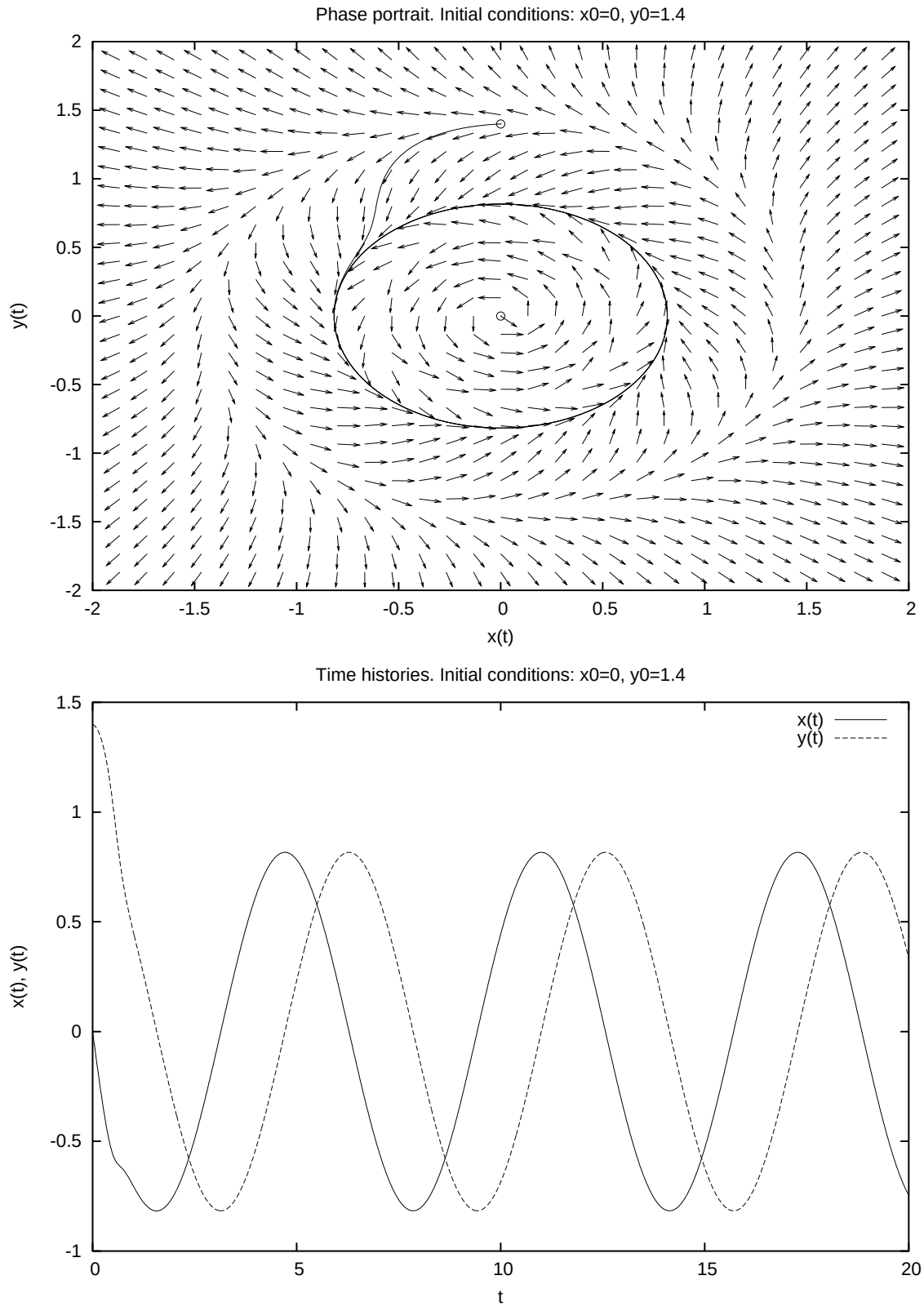


Figura 23: Ritratto di fase e storia temporale per $(x_0, y_0) = (0, 1.4)$ e $t_f = 20$.

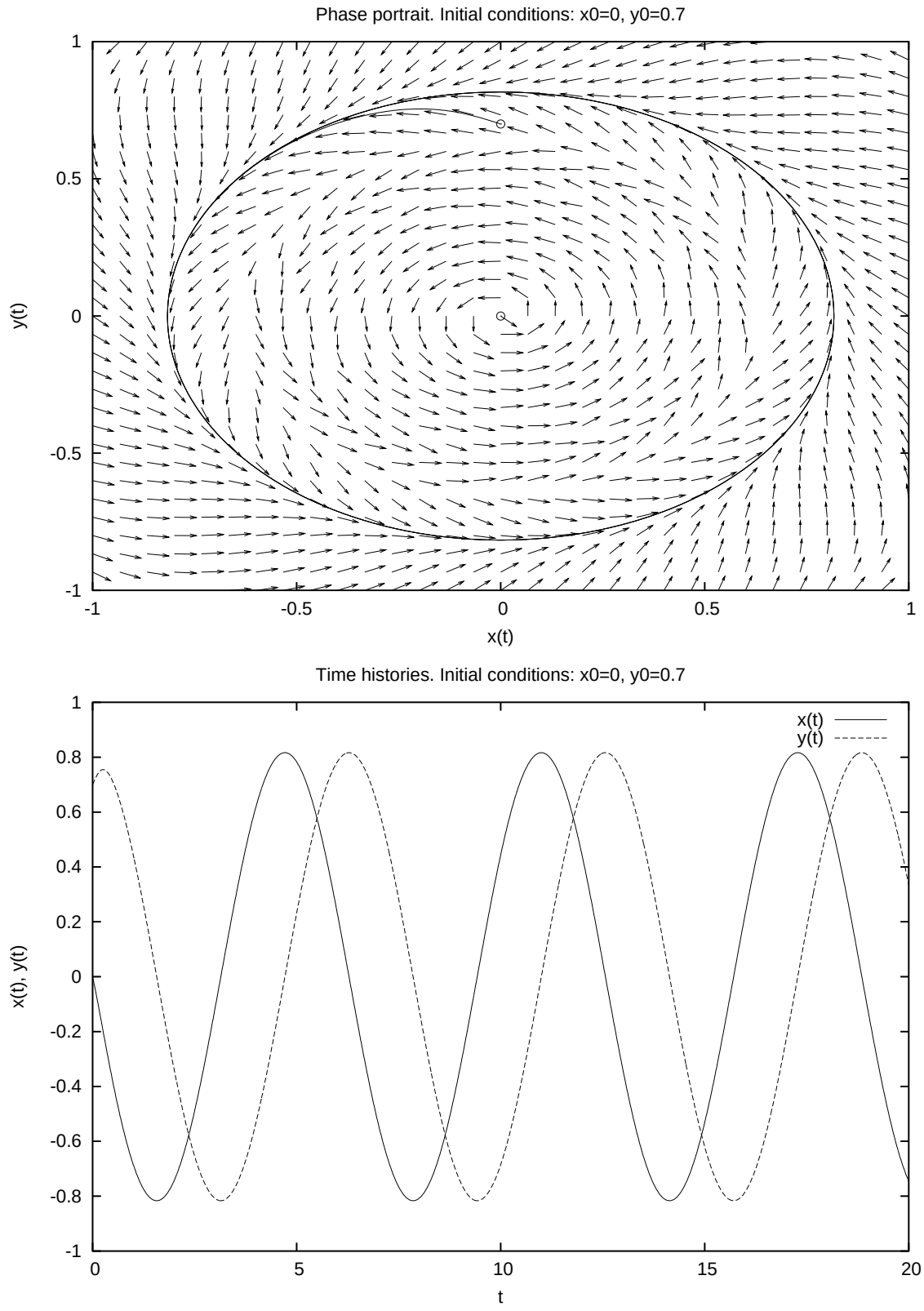


Figura 24: Ritratto di fase e storia temporale per $(x_0, y_0) = (0, 0.7)$ e $t_f = 20$.

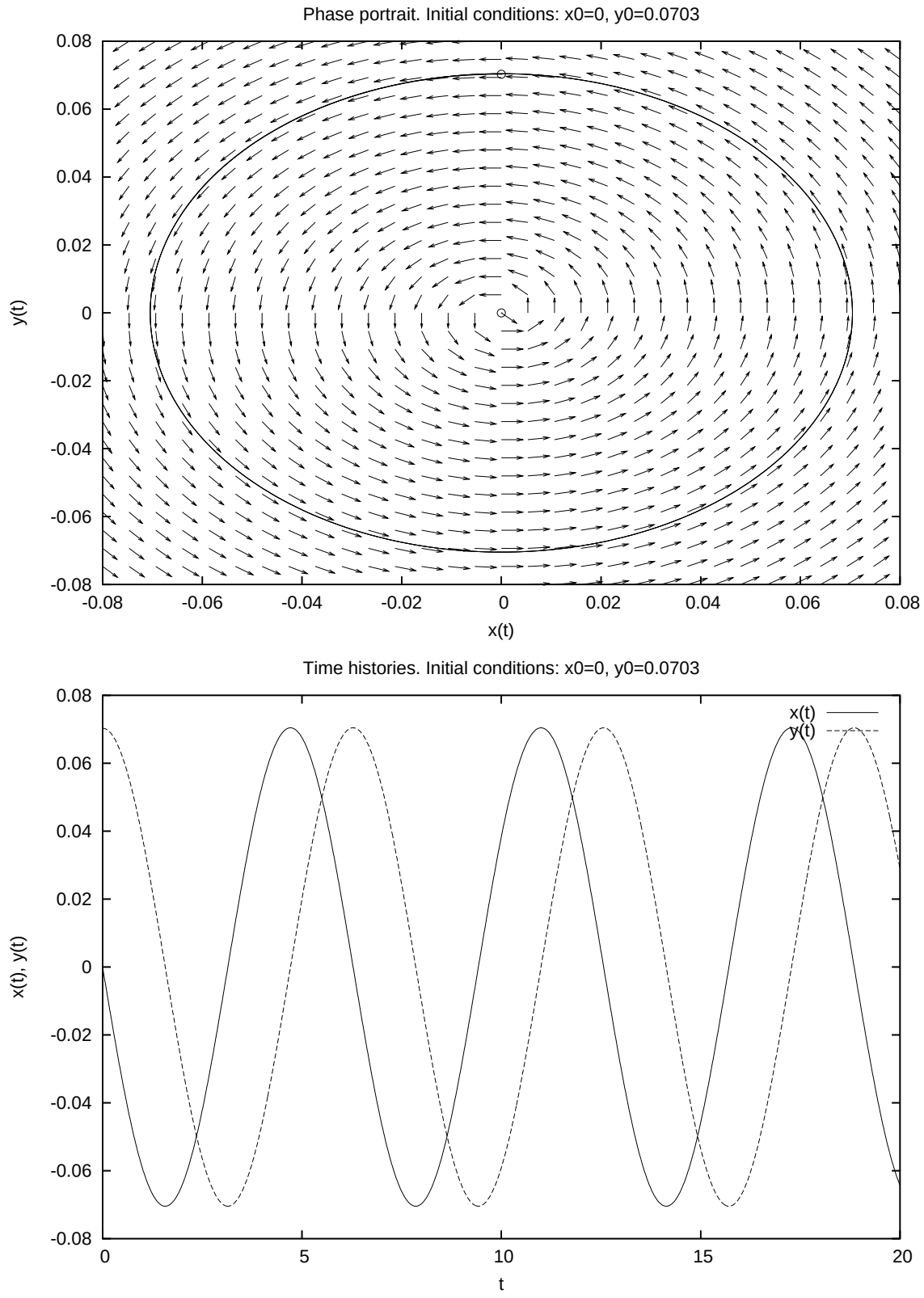


Figura 25: Ritratto di fase e storia temporale per $(x_0, y_0) = (0, 0.0703)$ e $t_f = 20$.

3.4 Esercizio

Studiare, al variare delle condizioni iniziali, del parametro $a > 0$ e del tempo finale, il comportamento dell'equazione (di van der Pool)

$$y'' + a(y^2 - 1)y' + y = 0.$$

3.4.1 Risoluzione

Si noti innanzitutto che l'equazione data può essere riscritta sotto forma di sistema del prim'ordine

$$\begin{cases} x' &= -y - a(y^2 - 1)x \\ y' &= x \end{cases}$$

L'unico punto di equilibrio per il sistema dato è l'origine. Lo Jacobiano è

$$J(x, y) = \begin{pmatrix} -a(y^2 - 1) & -(2axy + 1) \\ 1 & 0 \end{pmatrix},$$

che calcolato nell'origine risulta

$$J(0, 0) = \begin{pmatrix} a & -1 \\ 1 & 0 \end{pmatrix}$$

ed ammette gli autovalori $\lambda_{1,2} = (a \pm \sqrt{a^2 - 4})/2$. Essi sono complessi a parte reale positiva se $0 < a < 2$, pertanto in questo caso l'origine è un fuoco instabile (le orbite se ne allontanano seguendo una spirale). Se invece $a > 2$ l'origine è un nodo instabile essendo entrambi gli autovlori negativi. Si può dimostrare (attraverso l'analisi nel *piano di Lienard*) che esiste un unico ciclo stabile nel senso che ogni traiettoria non passante per l'origine si avvolge su di esso per $t \to +\infty$. Omettiamo qui la dimostrazione, ma procediamo con l'analisi di tipo numerico.

La figura 26 mostra la natura dell'origine, i.e. il suo essere un fuoco instabile ($a = 1$, $(x_0, y_0) = (10^{-4}, 10^{-4})$ e $t_f = 100$). Come si può facilmente intuire guardando il ritratto di fase, partendo da ogni punto interno al ciclo stabile, la traiettoria ne viene attratta. La figura 27 ($a = 1$, $(x_0, y_0) = (-2, 2.1)$ e $t_f = 100$) mostra invece come anche partendo da una condizione iniziale esterna al ciclo stabile, comunque per $t \to +\infty$ la traiettoria ne viene attratta. Infine, in figura 28 viene riportato un caso con $a = 0.1$. Come si può notare, cambiare il parametro a significa essenzialmente cambiare la forma del ciclo stabile ma non la sua natura.

Utilizzanto il file `sys4.m` riportato in appendice A si osservi il comportamento del sistema al variare dei parametri a disposizione.

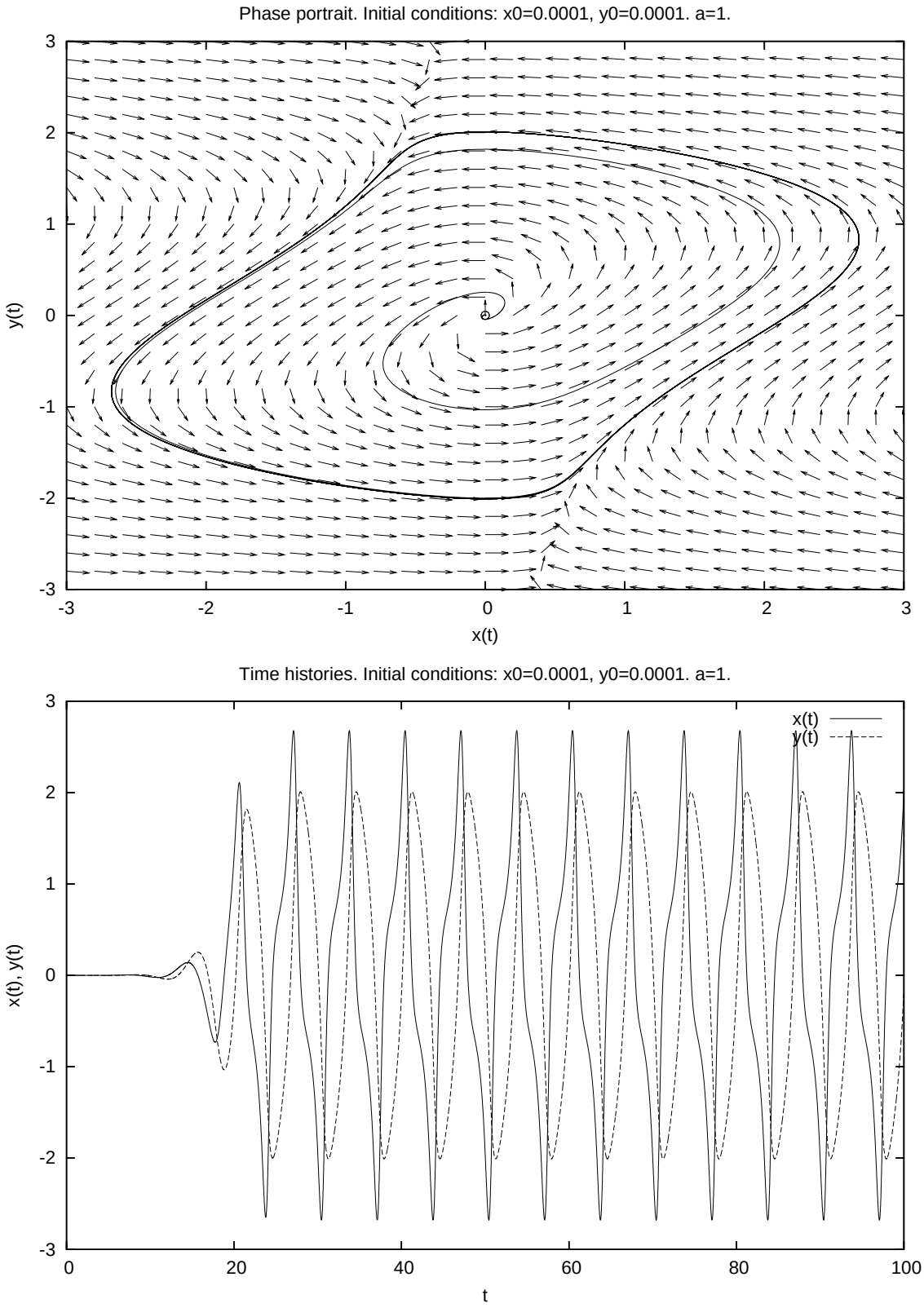


Figura 26: Ritratto di fase e storia temporale per $a = 1$, $(x_0, y_0) = (10^{-4}, 10^{-4})$ e $t_f = 100$.

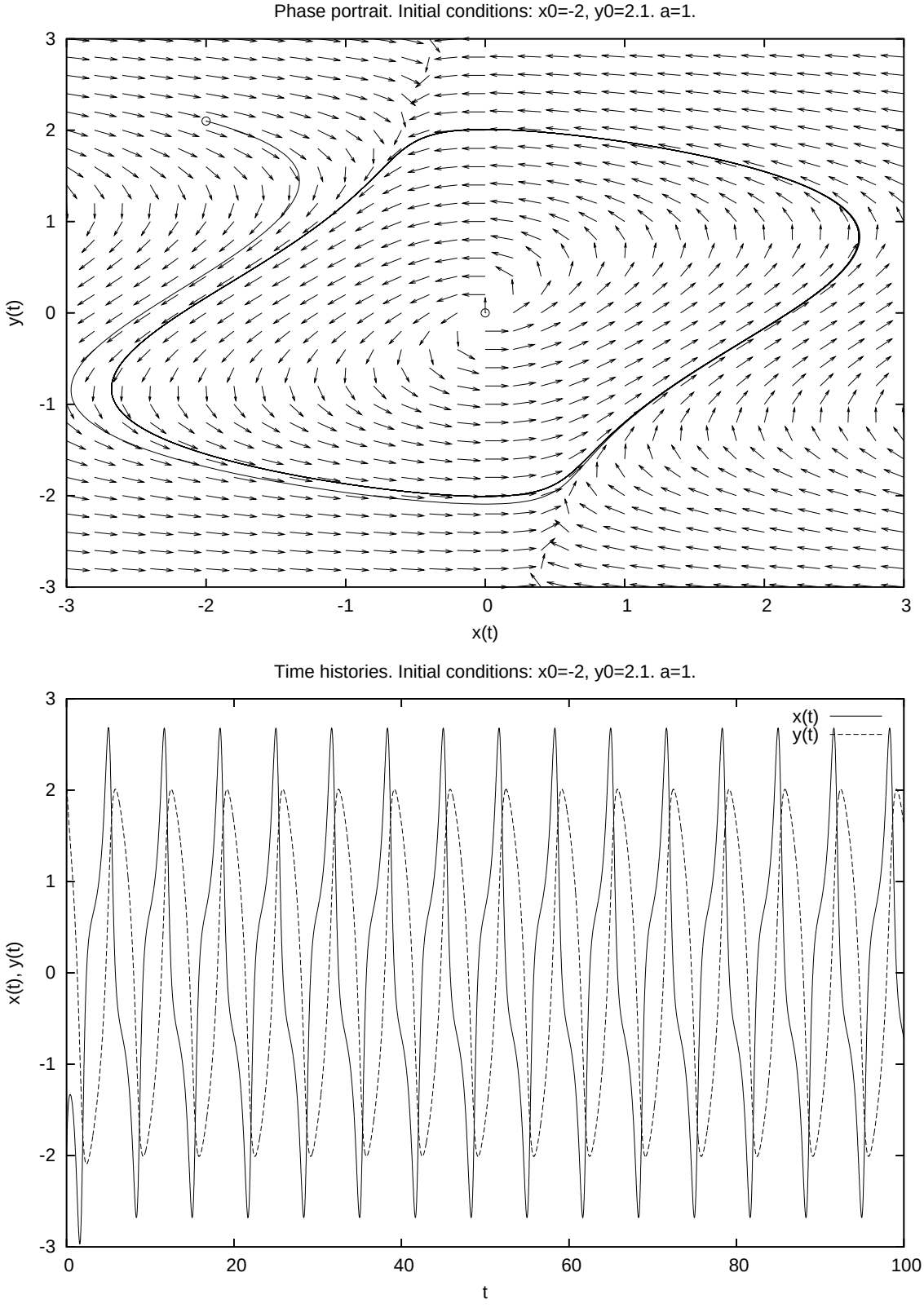


Figura 27: Ritratto di fase e storia temporale per $a = 1$, $(x_0, y_0) = (-2, 2.1)$ e $t_f = 100$.

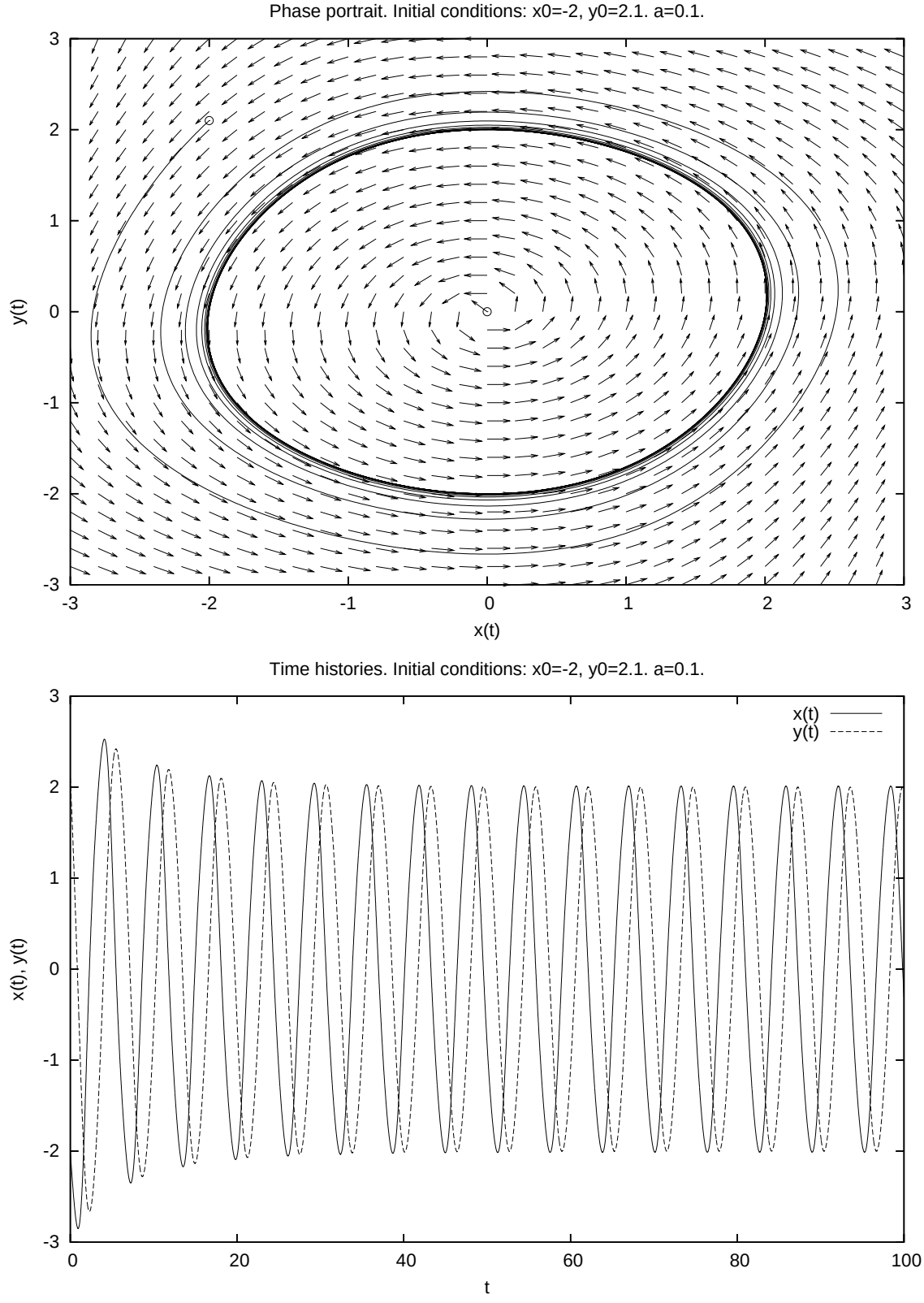


Figura 28: Ritratto di fase e storia temporale per $a = 0.1$, $(x_0, y_0) = (-2, 2.1)$ e $t_f = 100$.

3.5 Esercizio

Studiare, al variare delle condizioni iniziali e del tempo finale, il comportamento del sistema

$$\begin{cases} x' &= y \\ y' &= (1 - x^2 - y^2)y - x \end{cases}$$

3.5.1 Risoluzione

L'unico punto di equilibrio per il sistema dato è l'origine. Per capirne la natura si procede, come al solito, o tramite l'analisi degli autovalori del sistema linearizzato attorno all'origine oppure tramite lo studio dell'equazione differenziale ordinaria della distanza dall'origine ($\rho = \sqrt{x^2 + y^2}$) ricavata moltiplicando la prima equazione per x, la seconda per y, e sommandole. Così facendo si ottiene $\rho\,\rho' = y^2(1-\rho^2)$. Quindi, l'unica orbita per la quale $\rho' = 0$ è la circonferenza unitaria con centro nell'origine. La distanza dall'origine aumenta per condizioni iniziali interne a tale circonferenza e diminuisce per condizioni iniziali esterne. Pertanto, la circonferenza unitaria è un ciclo stabile e l'origine un fuoco instabile.

In file `sys5.m` può essere utilizzato per lo studio del sistema dato. Le figure 29 e 30 mostrano le caratteristiche sopra descritte.

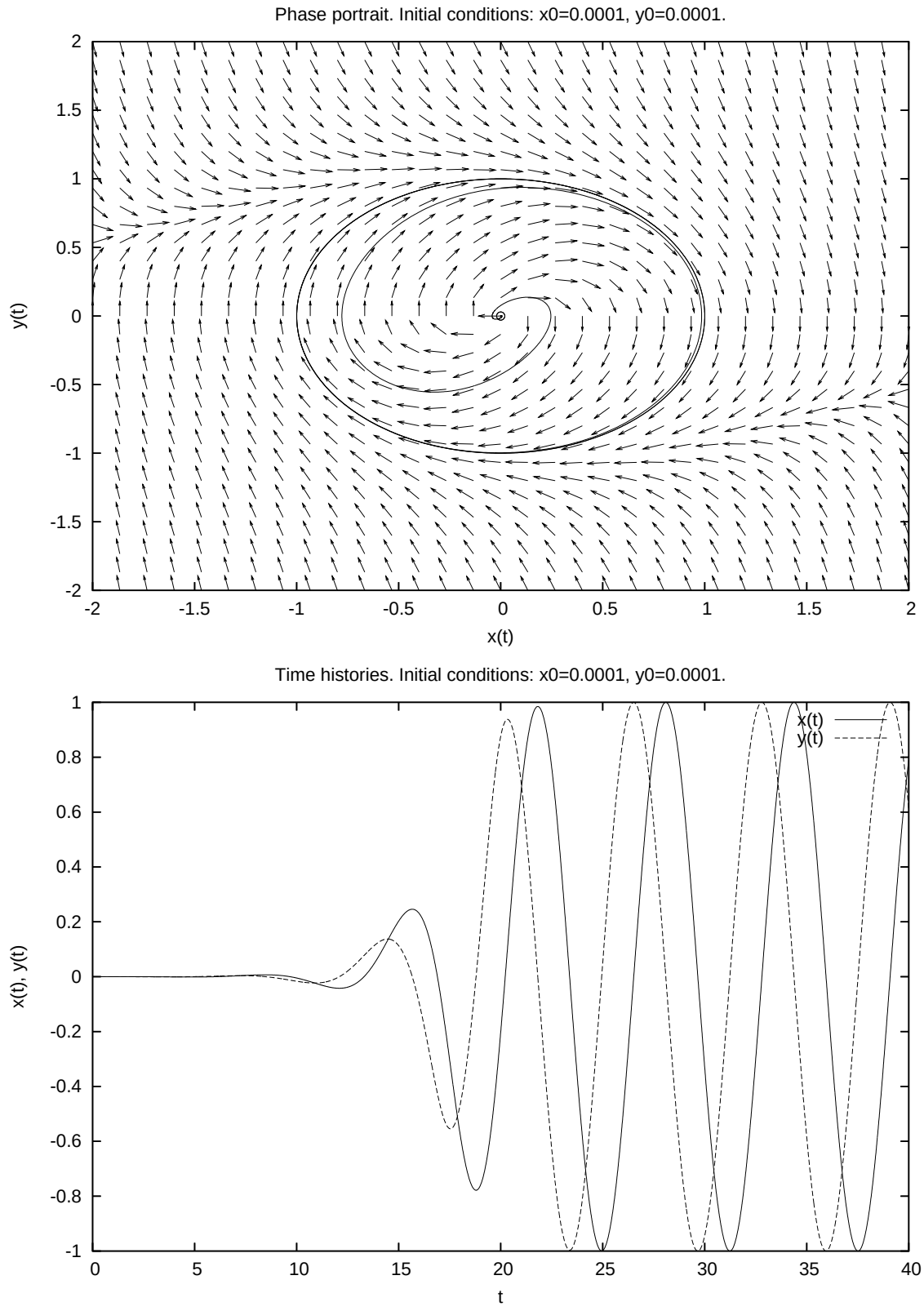


Figura 29: Ritratto di fase e storia temporale per $(x_0, y_0) = (10^{-4}, 10^{-4})$ e $t_f = 40$.

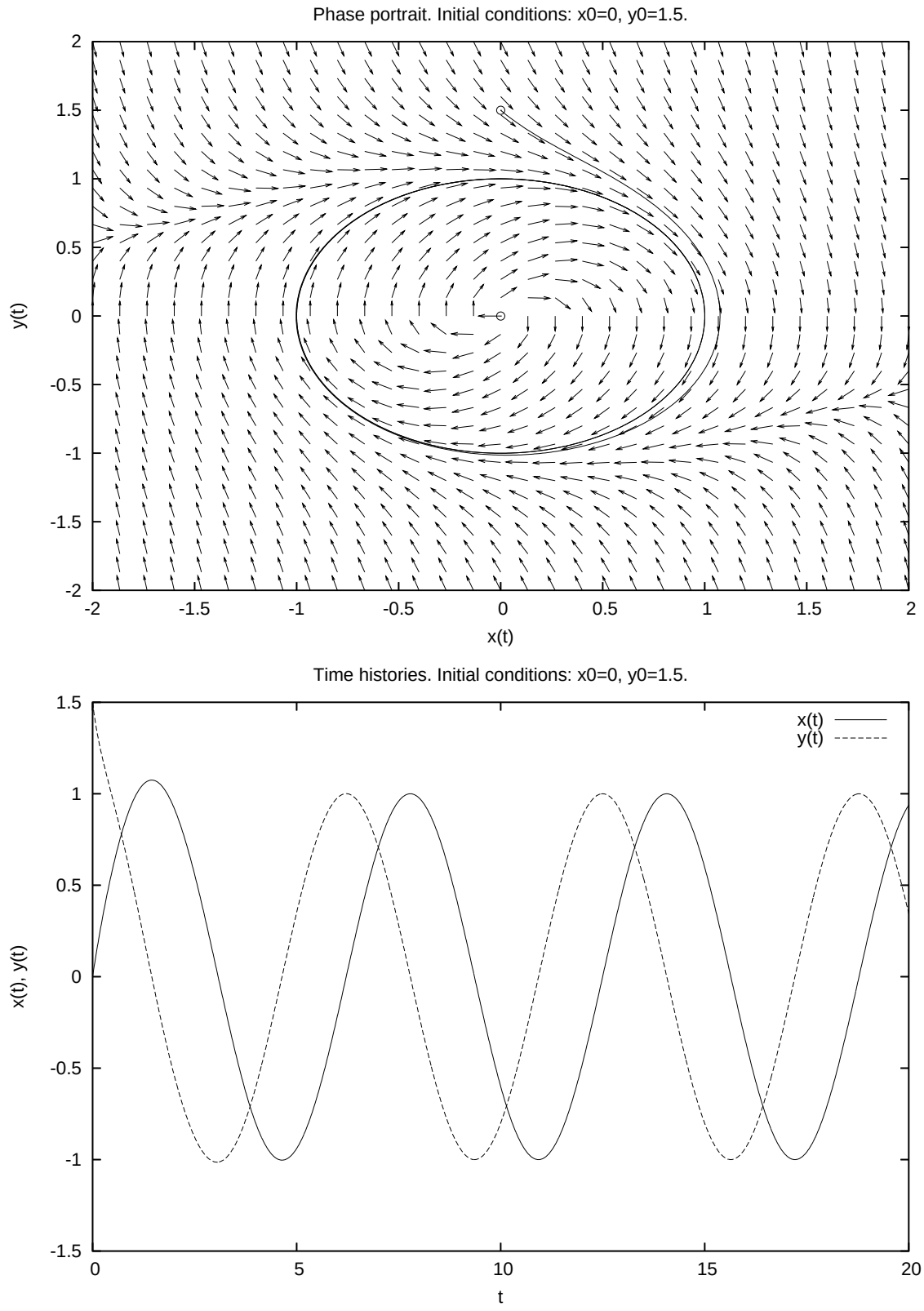


Figura 30: Ritratto di fase e storia temporale per $(x_0, y_0) = (0, 1.5)$ e $t_f = 20$.

3.6 Esercizio

Utilizzando le competenze acquisite in questa esercitazione e modificando adeguatamente gli script per Octave, studiare, al variare delle condizioni iniziali e del tempo finale, il comportamento dei sistemi

$$\begin{cases} x' &= x - y - x^3 \\ y' &= x - x^2 y \end{cases}$$

$$\begin{cases} x' &= -y^2 \\ y' &= -x^2 \end{cases}$$

4 Interazione tra due popolazioni (competizione, cooperazione ed esclusione competitiva)

4.1 Esercizio

Studiare, al variare dei parametri a, b, c, d (positivi), delle condizioni iniziali e del tempo finale, il comportamento del sistema

$$\begin{cases} x' &= ax - bxy \\ y' &= -cy + dxy \end{cases}$$

detto anche di Lotka-Volterra.

4.1.1 Risoluzione

Il sistema ammette due punti di equilibio, l'origine e $(c/d, a/b)$. Analizzando il sistema linearizzato attorno all'origine, si scopre che gli autovalori sono reali e $\lambda_1 = a, \lambda_2 = -c$. Pertanto, finché $a, c > 0$ l'origine è un punto di sella. La linearizzazione attorno al punto $(c/d, a/b)$ fornisce gli autovalori $\lambda_{1,2} = \pm i\sqrt{ac}$, pertanto finché $a, c > 0$ questo punto è stabile ma non asintoticamente. Infatti, partendo da qualsiasi condizione iniziale, il sistema evolve lungo traiettorie chiuse periodiche che riportano la soluzione a passare dalla condizione iniziale. Si osservi che una perturbazione nella soluzione porta il sistema su un'altra traiettoria chiusa e periodica che, per certi istanti di tempo, può trovarsi arbitrariamente lontana da quella imperturbata.

Utilizzando il file `myLV.m` riportato in appendice A si possono fare diverse prove al variare dei parametri a, b, c, d, della condizione iniziale e del tempo finale di integrazione. **Si ricordi di cambiare gli estremi della finestra di visualizzazione dipendentemente dalla condizione iniziale**.

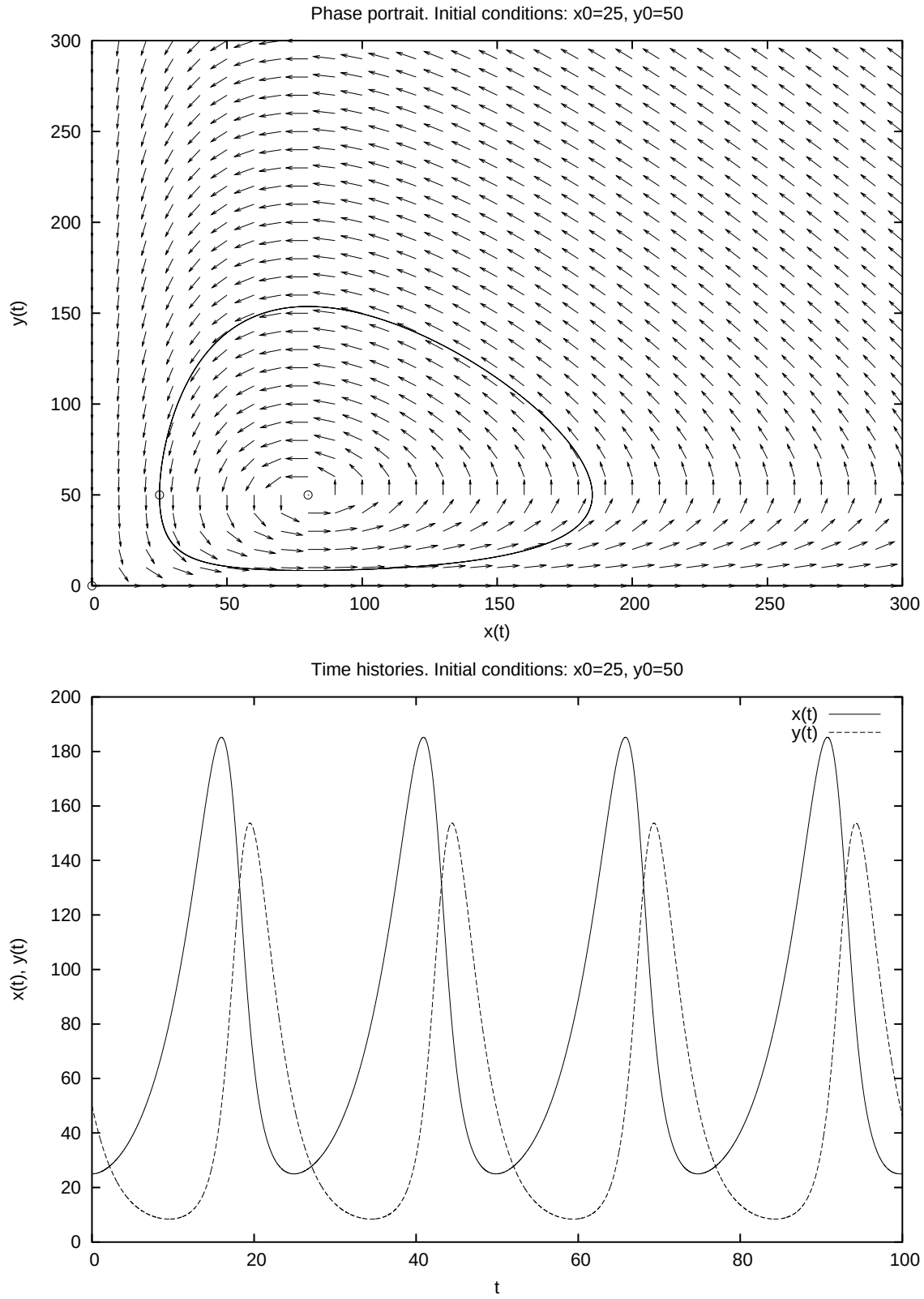


Figura 31: Ritratto di fase e storia temporale del sistema classico Lotka-Volterra per $a = 0.2, b = 0.004, c = 0.4, d = 0.005$, $(x_0, y_0) = (25, 50)$ e $t_f = 100$.

4.2 Esercizio

Studiare, al variare dei parametri $a, b \in \mathbb{R}$ (positivi e/o negativi), delle condizioni iniziali e del tempo finale, il comportamento del sistema

$$\begin{cases} x' &= x(1-x) - axy \\ y' &= y(1-y) - byx. \end{cases} \tag{2}$$

4.2.1 Risoluzione

Il sistema dato può essere riscritto come

$$\begin{cases} x' &= x - x^2 - axy \\ y' &= y - y^2 - byx, \end{cases}$$

e pertanto è un caso particolare del sistema generale

$$\begin{cases} x' &= \alpha_1 x + a_{11}x^2 + a_{12}xy \\ y' &= \alpha_2 y + a_{21}xy + a_{22}y^2 \end{cases} \tag{3}$$

con $\alpha_1 = \alpha_2 = 1$, $a_{11} = a_{22} = -1$ e $a_{12} = -a$, $a_{21} = -b$. Le due popolazioni hanno una legge di crescita molto simile, con un tasso di crescita positivo e pari a 1 (ossia entrambe possono esistere anche da sole), e un tasso di competizione *intraspecifica* pari a 1 ($-1 < 0$ significa che c'è competizione all'interno della stessa specie). Se a e b sono diversi da zero c'è anche un'interazione tra le due specie, detta *interspecifica*, che può essere di competizione (a, b entrambi positivi) o cooperazione (a, b entrambi negativi). Per lo studio numerico del sistema generale (3) utilizziamo lo script `compcoop.m`, riportato in appendice A.

I punti di equilibrio sono $(0,0)$, $(1,0)$, $(0,1)$ e $(\frac{a-1}{a\,b-1}, \frac{b-1}{a\,b-1})$.

Esaminiamo dapprima il caso $a = b = 0$, ovvero assenza di competizione/cooperazione. I punti di equilibrio sono i 4 vertici del quadrato di lato 1, di cui solo il punto $(1,1)$ è un pozzo (quindi stabile), mentre gli altri tre punti sono o sorgente (e quindi instabile, l'origine) o di sella ($(1,0)$ e $(0,1)$) e quindi comunque instabili. Partendo da condizioni iniziali positive per entrambe le specie, pertanto, il sistema evolverà verso il punto di equilibrio $(1,1)$. Questi risultati sono riassunti visivamente in figura 32.

Caso $0 < a = b < 1$. Il sistema è competitivo con uguale competizione per entrambe le specie. Il punto $(\frac{a-1}{a\,b-1}, \frac{b-1}{a\,b-1})$ è l'unico stabile ed attrae le soluzioni con condizione iniziale positiva per entrambe le specie. Questo è chiaramente visibile in figura 33.

Caso $a = b = 1$. Il punto $(\frac{a-1}{a\,b-1}, \frac{b-1}{a\,b-1})$ non esiste e gli altri tre sono comunque instabili. La soluzione è attratta dai punti di intersezione tra la retta $y = -x+1$ e la retta $y = \frac{y_0}{x_0}x$, dove (x_0, y_0) è la condizione iniziale. Questa situazione è visibile in figura 34, ottenuta per $a = b = 1$.

Caso $a = b > 1$. Il punto $(\frac{a-1}{a\,b-1}, \frac{b-1}{a\,b-1})$ diventa di sella, e quindi instabile. Al contrario, i punti $(1,0)$ e $(0,1)$ diventano dei fuochi stabili, come visibile in figura 35 per $a = b = 1.5$. Pertanto la competizione è tale da far sopravvivere solo la specie che parte avvantaggiata (si vedano le zone del bacino di attrazione dei fuochi stabili). L'origine rimane una sorgente (instabile). È chiaro che, partendo da condizioni iniziali $0 < x_0 = y_0 < 1$ la soluzione finisce nel punto $(\frac{a-1}{a\,b-1}, \frac{b-1}{a\,b-1})$, che è instabile essendo di sella

Caso $a = b >> 1$. Il punto $(\frac{a-1}{a\,b-1}, \frac{b-1}{a\,b-1})$ tende all'origine e, come visto nell'esempio precedente, una delle due popolazioni si estingue molto velocemente (vedi figura 36). Plottanto il prodotto $\varphi(t) = x(t)y(t)$ si può avere un'idea del tasso di segregazione delle popolazioni.

Caso $-1 < a = b < 0$. Il sistema è cooperativo, nel senso che le due specie anziché contrastarsi vicendevolmente, cooperano alla sopravvivenza l'una dell'altra. In particolare, il punto $(\frac{a-1}{a\,b-1}, \frac{b-1}{a\,b-1})$ è l'unico stabile ed attrae tutte le traiettorie (purché non partano da punti di equilibrio), come si vede in figura 37.

Caso $a = b-1$. Il punto $(\frac{a-1}{a\,b-1}, \frac{b-1}{a\,b-1})$ non esiste e gli altri punti di equilibrio sono instabili. Pertanto, entrambe le soluzioni crescono indefinitamente (il sistema è cooperativo), come chiaramente provato dalla figura 38.

Caso $a = b < -1$. Il punto $(\frac{a-1}{a\,b-1}, \frac{b-1}{a\,b-1})$ si trova nel terzo quadrante e quindi non viene considerato. Gli altri tre sono instabili, per cui entrambe le soluzioni crescono indefinitamente in tempi molto brevi (vedi figura 39).

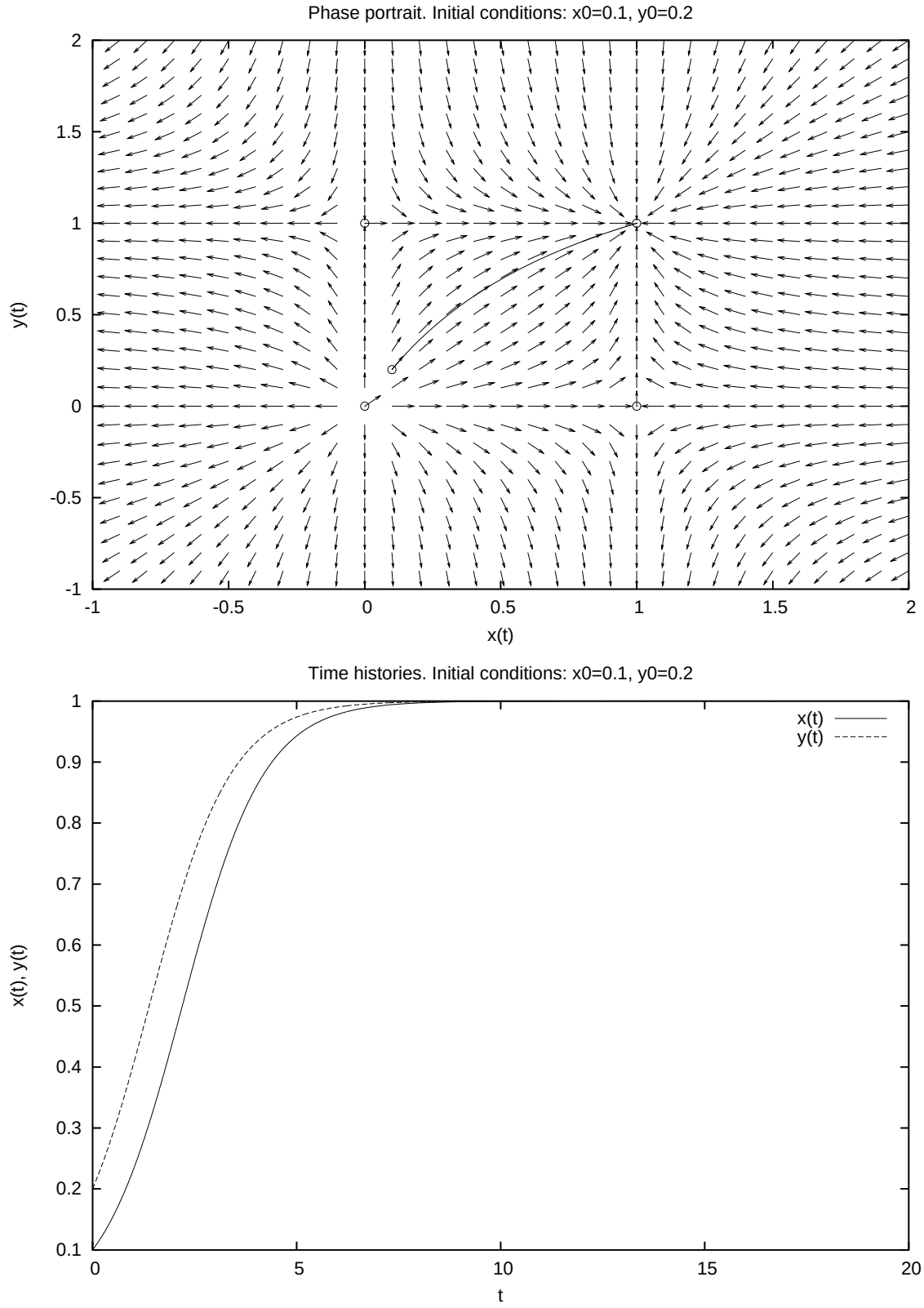


Figura 32: Ritratto di fase e storia temporale per $a = b = 0$, $(x_0, y_0) = (0.1, 0.2)$ e $t_f = 20$.

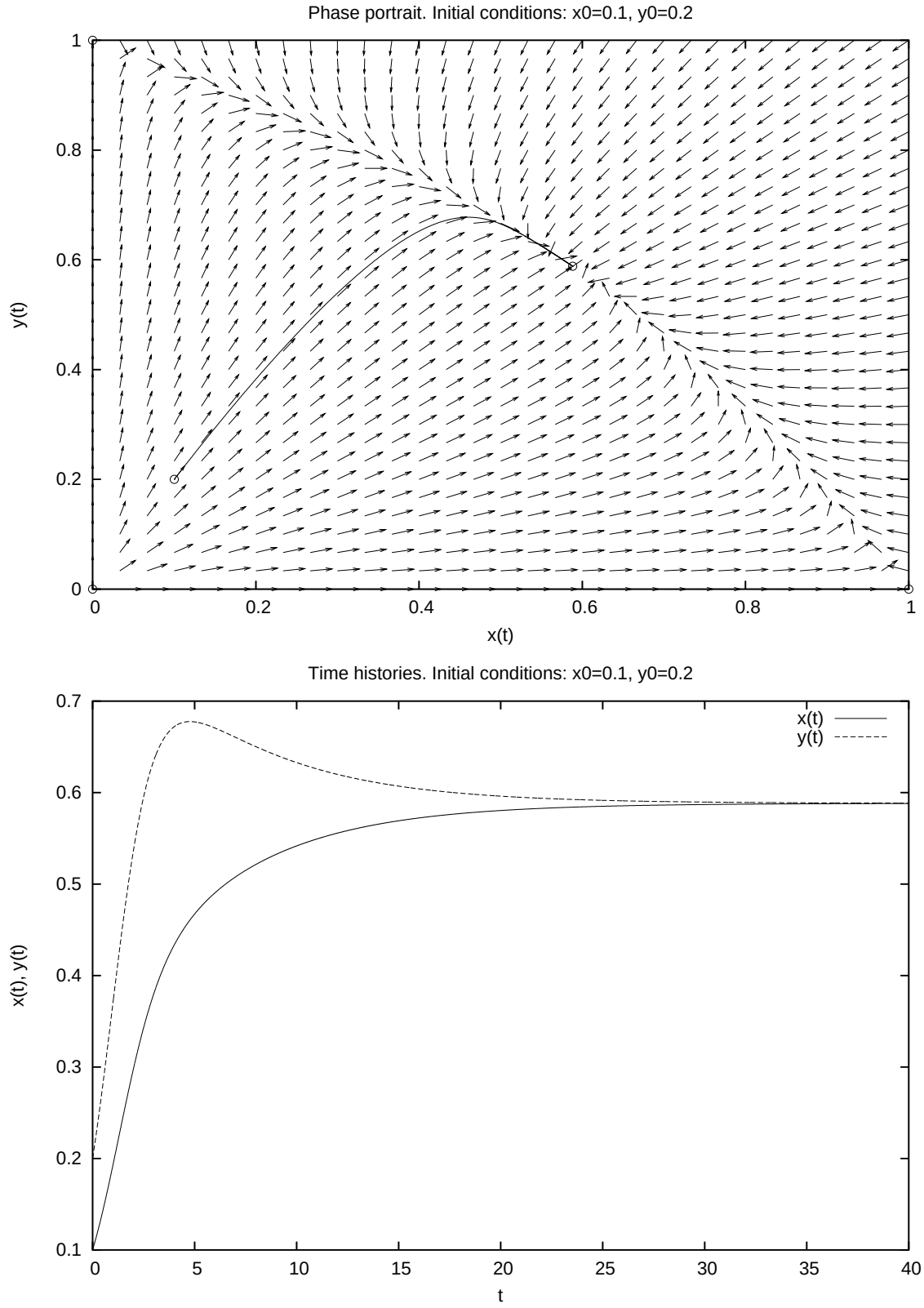


Figura 33: Ritratto di fase e storia temporale per $a = b = 0.7$, $(x_0, y_0) = (0.1, 0.2)$ e $t_f = 40$.

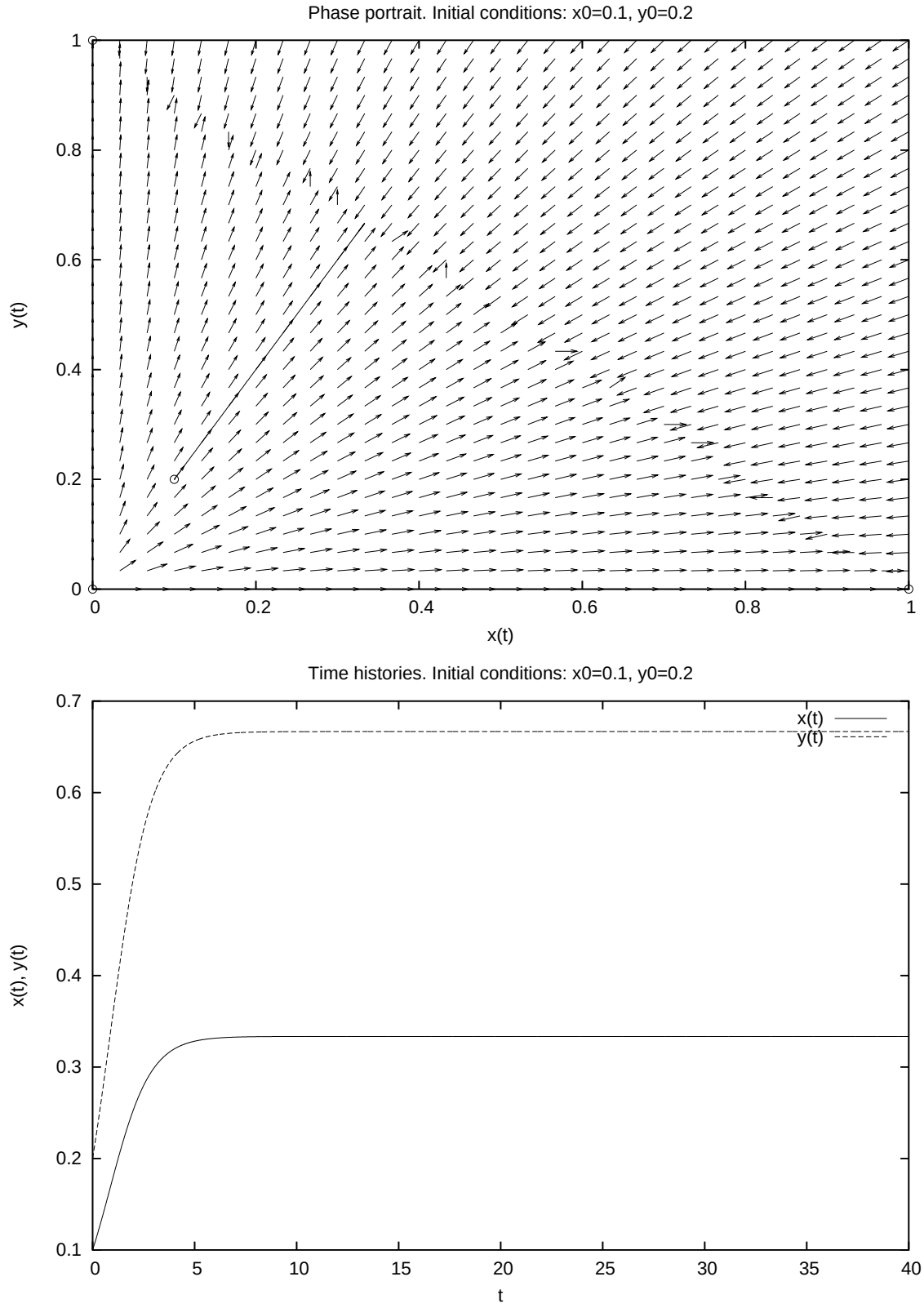


Figura 34: Ritratto di fase e storia temporale per $a = b = 1$, $(x_0, y_0) = (0.1, 0.2)$ e $t_f = 40$.

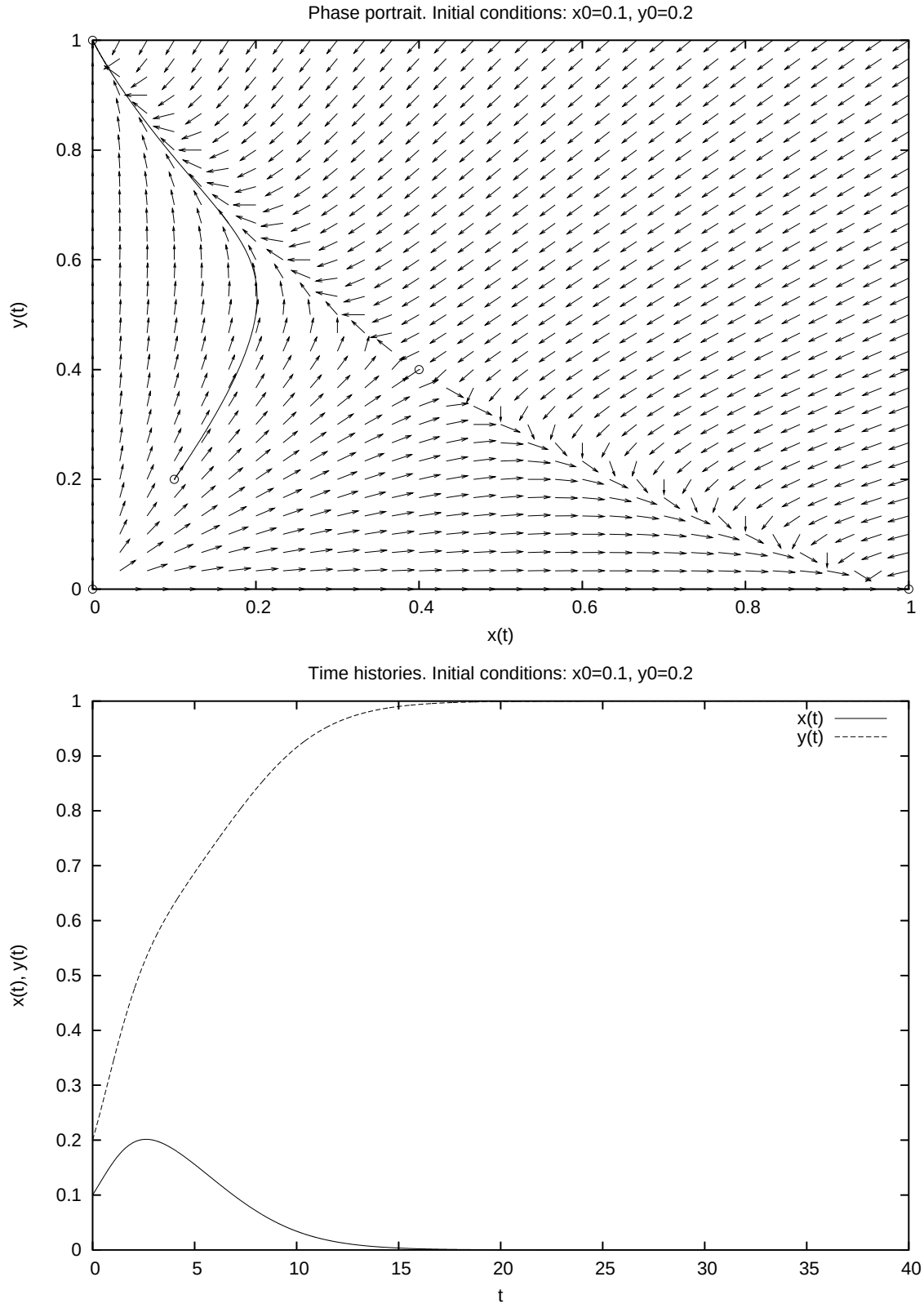


Figura 35: Ritratto di fase e storia temporale per $a = b = 1.5$, $(x_0, y_0) = (0.1, 0.2)$ e $t_f = 40$.

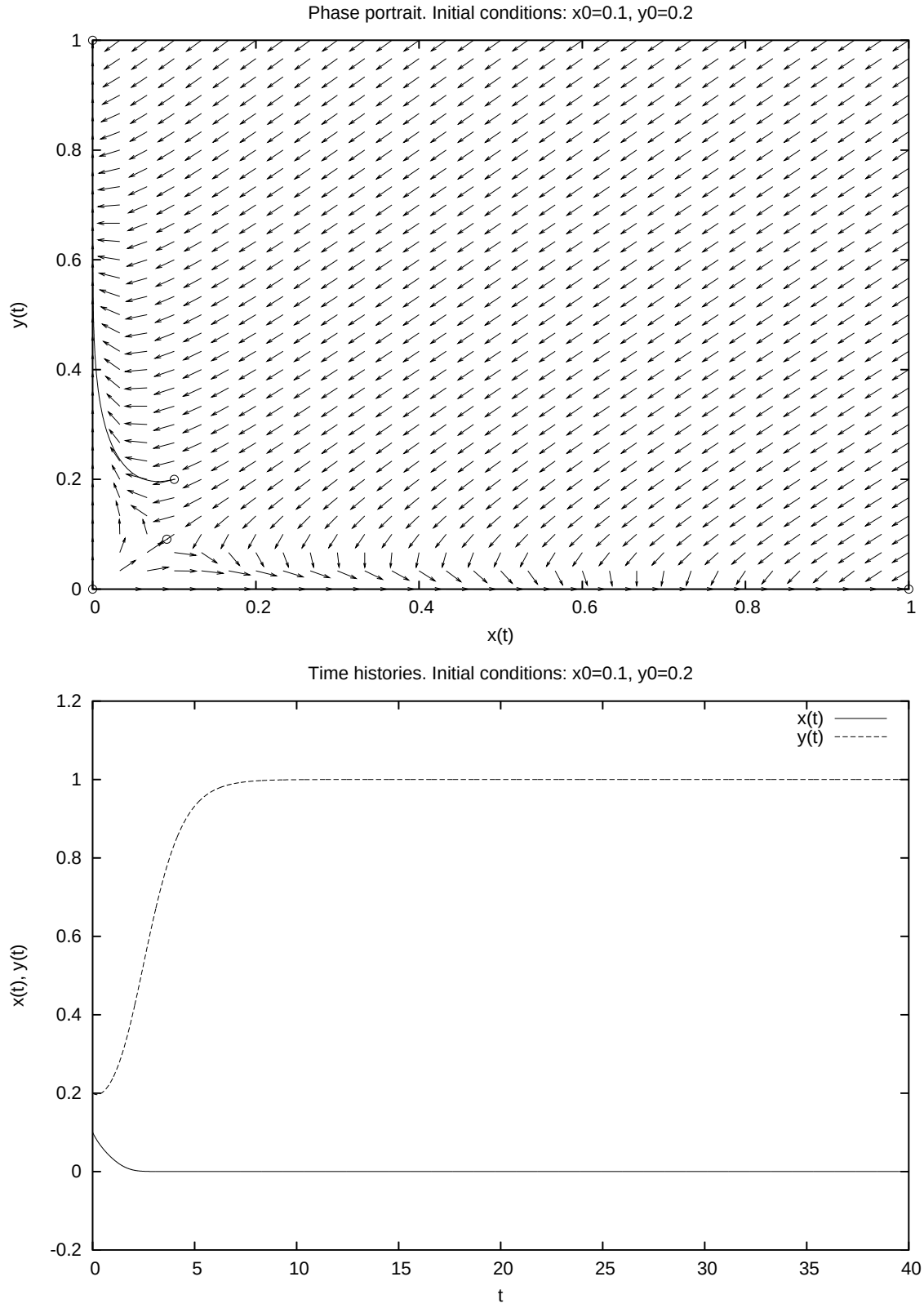


Figura 36: Ritratto di fase e storia temporale per $a = b = 10$, $(x_0, y_0) = (0.1, 0.2)$ e $t_f = 40$.

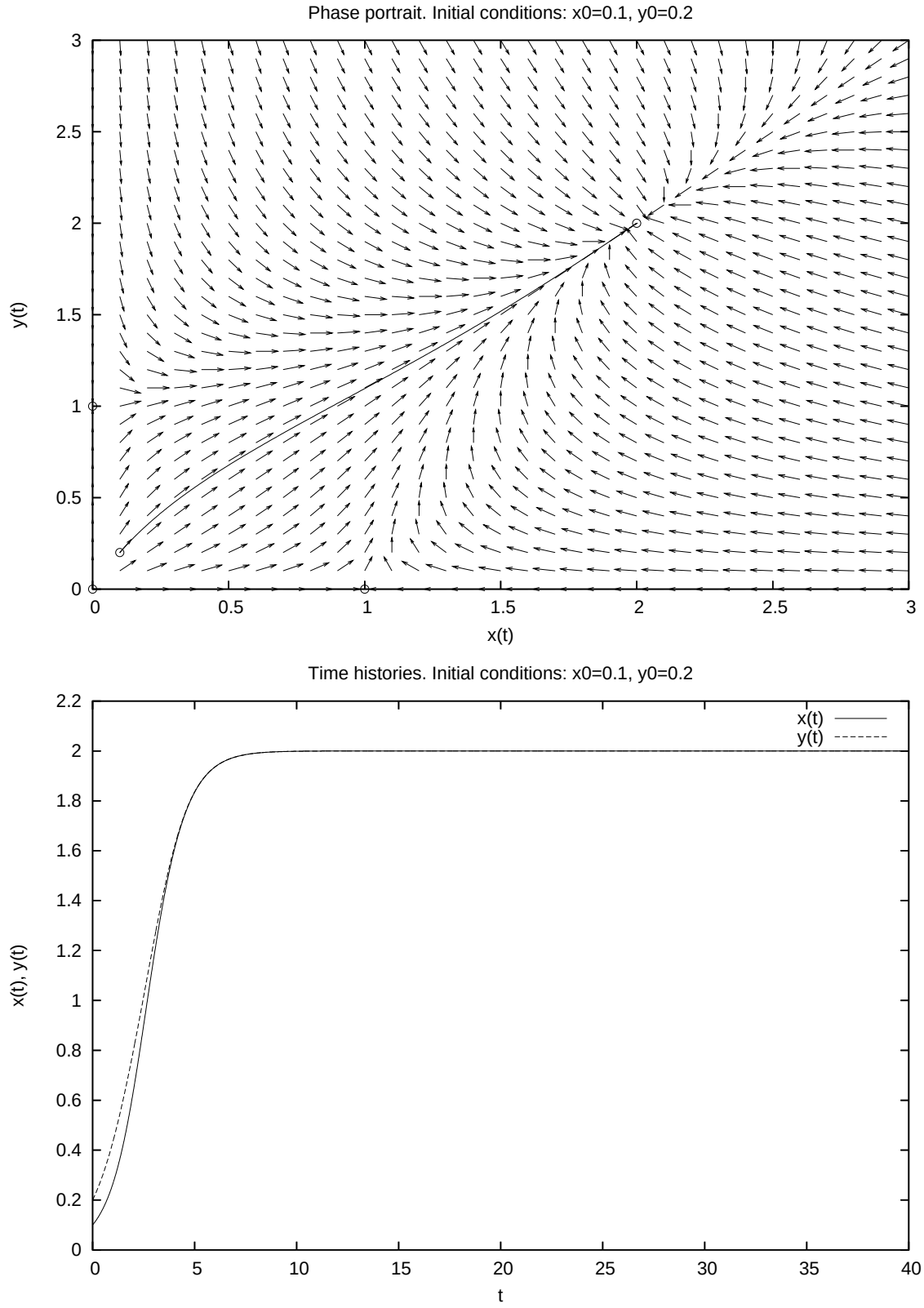


Figura 37: Ritratto di fase e storia temporale per $a = b = -0.5$, $(x_0, y_0) = (0.1, 0.2)$ e $t_f = 40$.

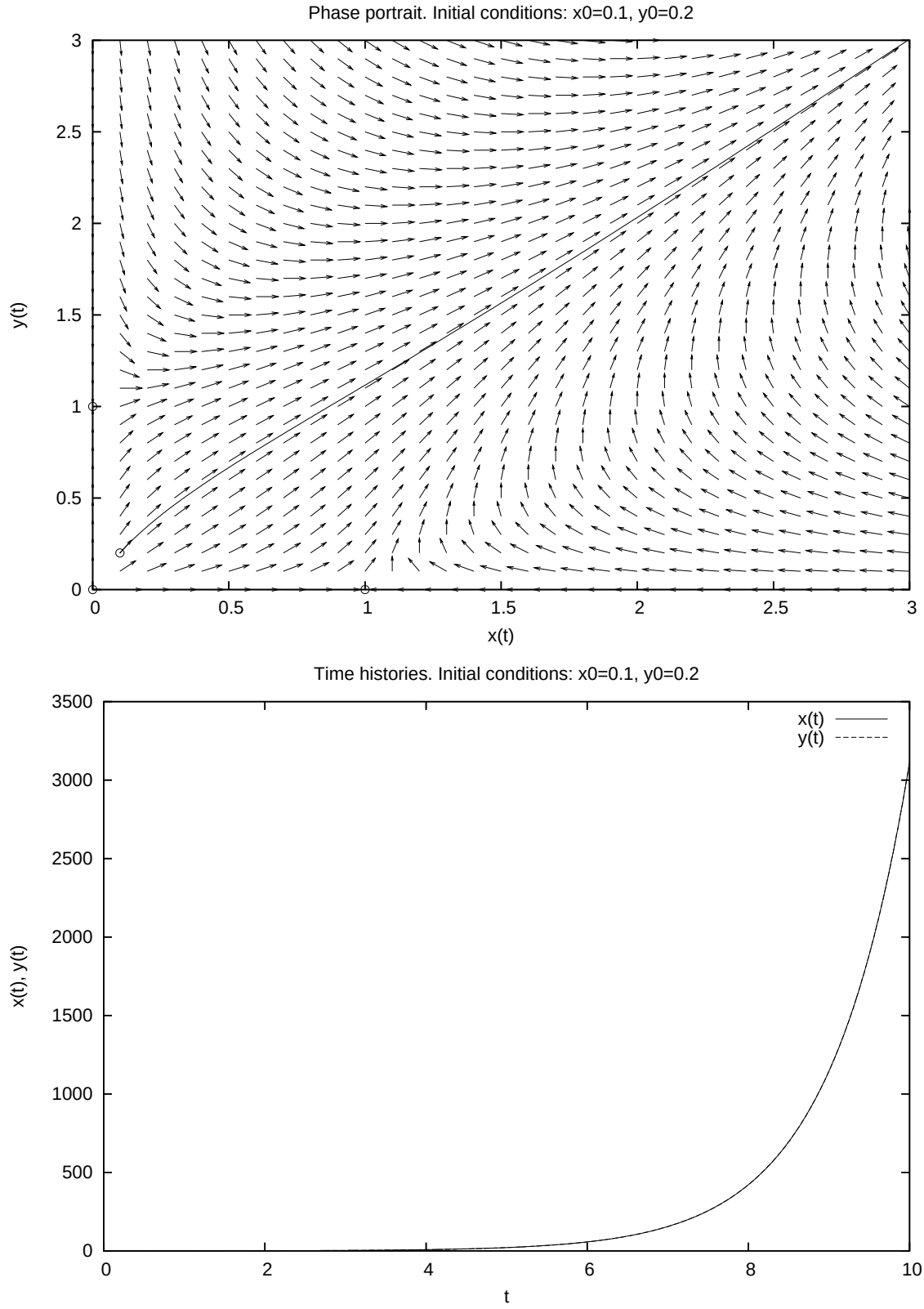


Figura 38: Ritratto di fase e storia temporale per $a = b = -1.0$, $(x_0, y_0) = (0.1, 0.2)$ e $t_f = 10$.

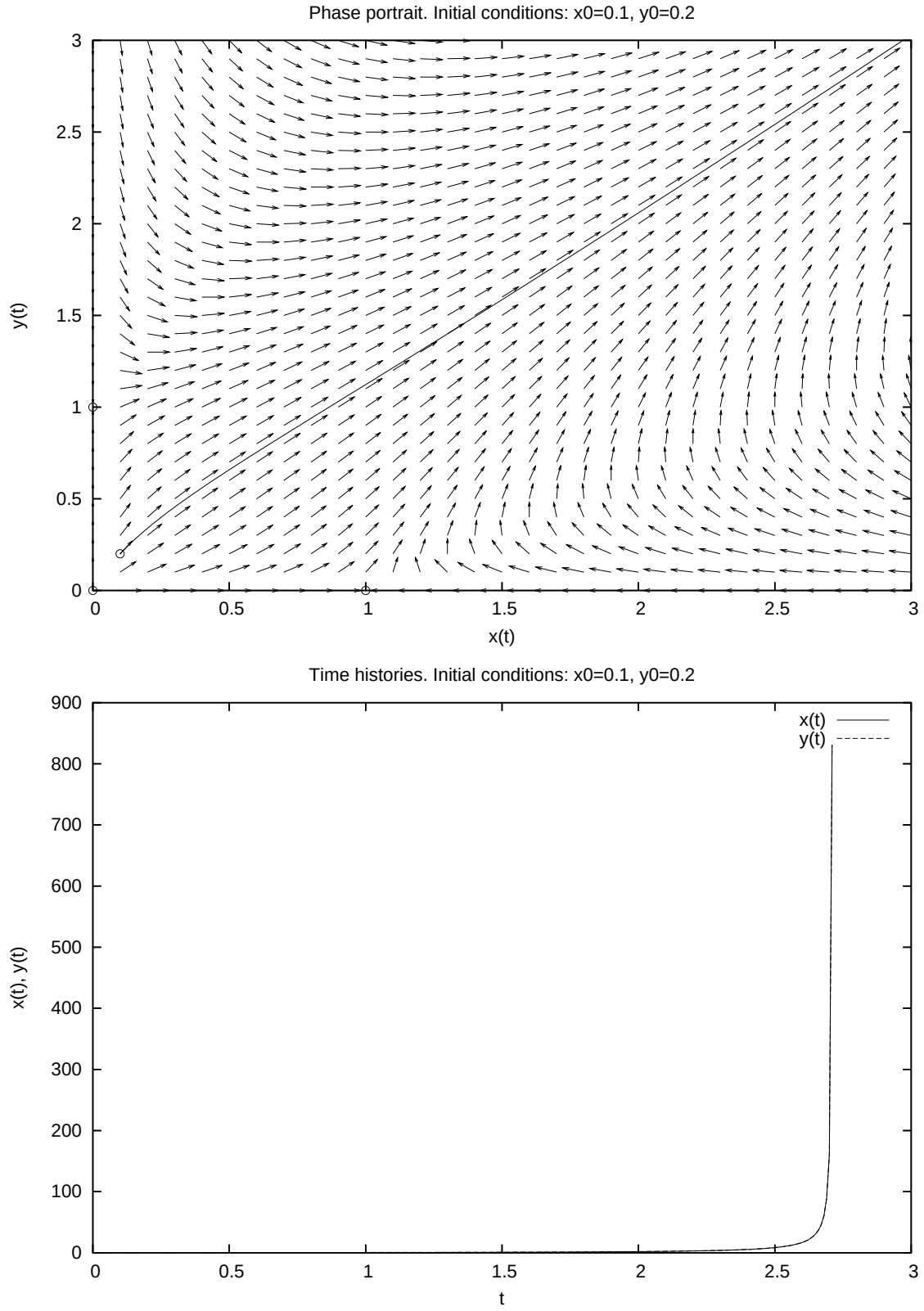


Figura 39: Ritratto di fase e storia temporale per $a = b = -1.5$, $(x_0, y_0) = (0.1, 0.2)$ e $t_f = 2.71$.

4.3 Esercizio

Studiare, al variare del parametro k (positivo), delle condizioni iniziali e del tempo finale, il comportamento del sistema

$$\begin{cases} x' &= x(1-x) - kxy^2 \\ y' &= y(1-y) - kyx^2. \end{cases} \tag{4}$$

4.3.1 Risoluzione

Il sistema dato è simile al sistema 2 con la differenza che ora l'accoppiamento tra le due specie non è più quadratico ma cubico. Si noti che, essendo $k > 0$, le due specie sono in competizione tra loro. I punti di equilibrio sono i seguenti: $(0,0)$, $(1,0)$, $(0,1)$, $(-\frac{\sqrt{4k+1}+1}{2k}, -\frac{\sqrt{4k+1}+1}{2k})$, $(\frac{\sqrt{4k+1}-1}{2k}, \frac{\sqrt{4k+1}-1}{2k})$, $(\frac{\sqrt{4k-3}+1}{2k}, -\frac{\sqrt{4k-3}-1}{2k})$, $(-\frac{\sqrt{4k-3}-1}{2k}, \frac{\sqrt{4k-3}+1}{2k})$. Ovviamente, trattandosi di popolazioni, consideriamo solo i punti di equilibrio nel primo quadrante, ovvero $(0,0)$, $(1,0)$, $(0,1)$ e $(\frac{\sqrt{4k+1}-1}{2k}, \frac{\sqrt{4k+1}-1}{2k})$. Anziché procedere attraverso lo studio analitico della stabilità dei punti di equilibrio, ci limitiamo allo studio numerico tramite il il file `LVmod.m` deducendo da esso le proprietà del sistema.

Consideriamo il caso $k > 1$, quindi per fissare le idee $k = 2$, e lasciamo allo studente lo studio per $0 < k \leq 1$. Come si può notare dalla figura 40, l'origine non è un punto di equilibrio stabile, mentre lo sono i punti $(1,0)$ e $(0,1)$. Il punto $(\frac{\sqrt{4k+1}-1}{2k}, \frac{\sqrt{4k+1}-1}{2k})$ è di sella. Analizzando il ritratto di fase, pertanto, si osserva che il piano $[0,1] \times [0,1]$ viene suddiviso in 4 regioni, due delle quali sono il bacino di attrazione del punto $(0,1)$ (quelle sotto la bisettrice del primo quadrante) e le altre due sono il bacino di attrazione del punto $(1,0)$ (le due sopra la bisettrice). Il fatto che questi due siano i soli punti di equilibrio del sistema ha una fortissima implicazione dal punto di vista della crescita delle popolazioni interagenti. Infatti, una è sempre destinata ad estinguersi (la minore delle due nella condizione iniziale) mentre l'altra raggiunge il massimo (la maggiore delle due nella condizione iniziale).

Partendo da una condizione iniziale sulla bisettrice, invece, la soluzione viene attratta dal punto di equilibrio $(\frac{\sqrt{4k+1}-1}{2k}, \frac{\sqrt{4k+1}-1}{2k})$, come mostrato nelle figure 41 e 42. Tuttavia, se la simulazione è estesa a tempi più lunghi, per esempio $t_f = 100$, si nota che la soluzione non rimane in $(\frac{\sqrt{4k+1}-1}{2k}, \frac{\sqrt{4k+1}-1}{2k})$ ma viene attratta da $(1,0)$ o $(0,1)$ dipendentemente dagli errori numerici (lo studente diligente lo provi).

Aumentando il valore di k, che rappresenta il coefficiente di competizione tra le due specie, il punto di equilibrio $(\frac{\sqrt{4k+1}-1}{2k}, \frac{\sqrt{4k+1}-1}{2k})$ si sposta verso l'origine (a cui tende per $k \to \infty$). Questo ha come effetto, a parità di condizione iniziale, un'aumentata velocità di convergenza verso i fuochi stabili. Confrontando i risultati per $k = 20$ (riportati in figura 43) con i risultati ottenuti per $k = 2$ (vedi figura 40), si osserva infatti che la specie che tende a zero ci tende molto più velocemente nel caso di k elevato.

Per avere un'idea di quanto velocemente una delle due popolazioni si estingue, basta plottare il prodotto xy.

Nota. Alla luce dei risultati ottenuti in questo esercizio, dovrebbe essere più immediato interpretare quanto trovato nell'esercizio precedente.

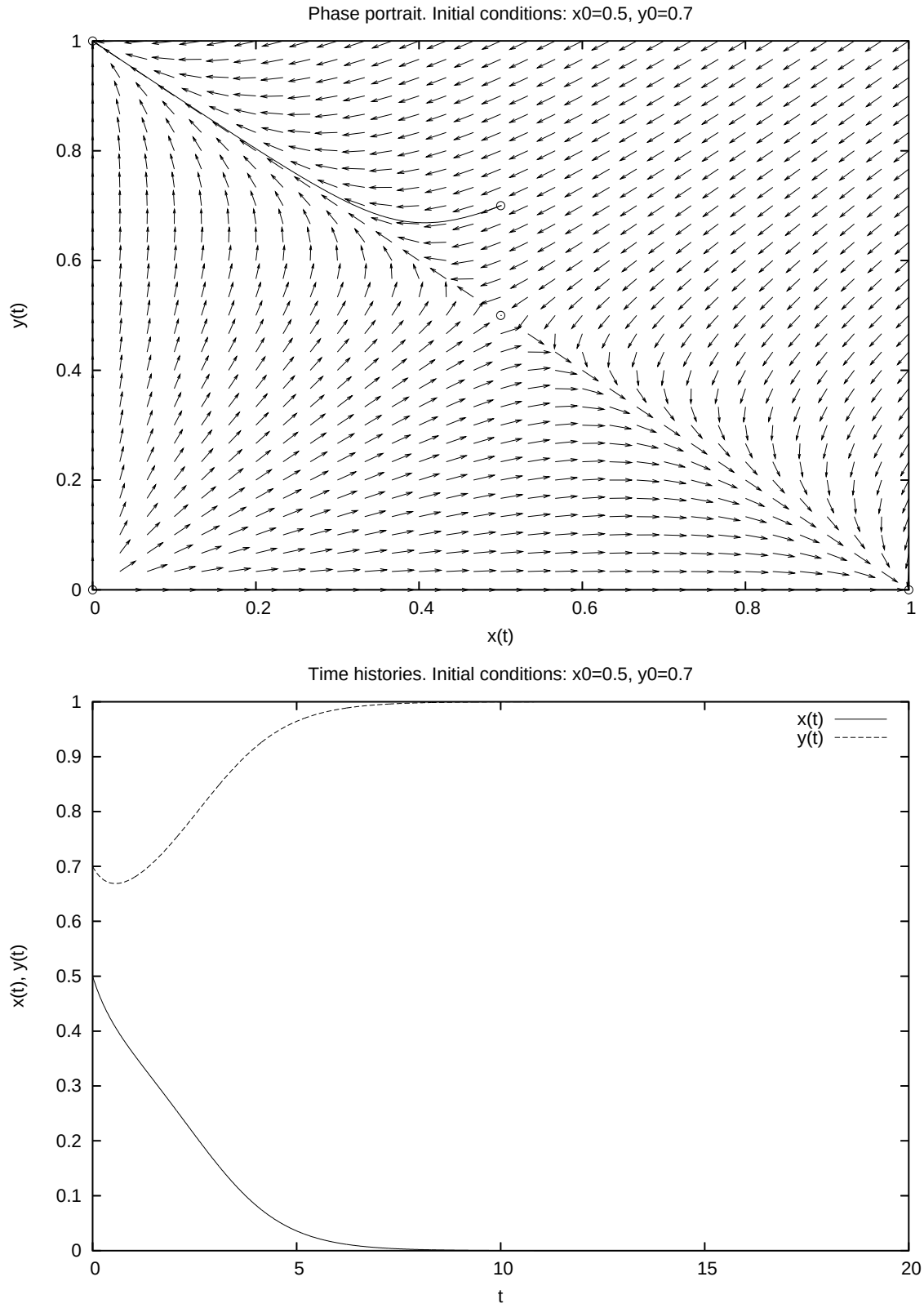


Figura 40: Ritratto di fase e storia temporale del sistema Lotka-Volterra modificato per $k = 2$, $(x_0, y_0) = (0.5, 0.7)$ e $t_f = 20$.

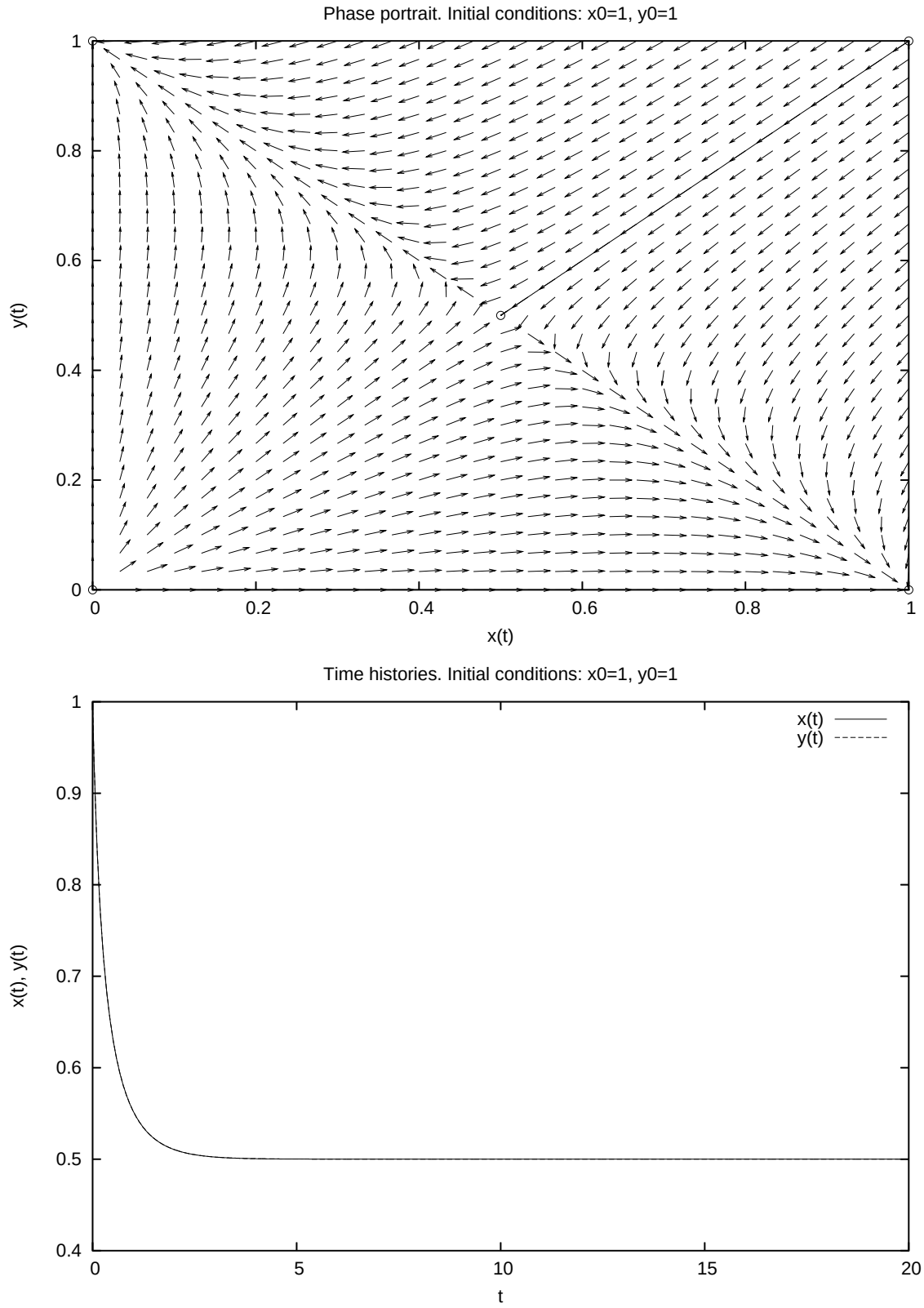


Figura 41: Ritratto di fase e storia temporale del sistema Lotka-Volterra modificato per $k = 2$, $(x_0, y_0) = (1.0, 1.0)$ e $t_f = 20$.

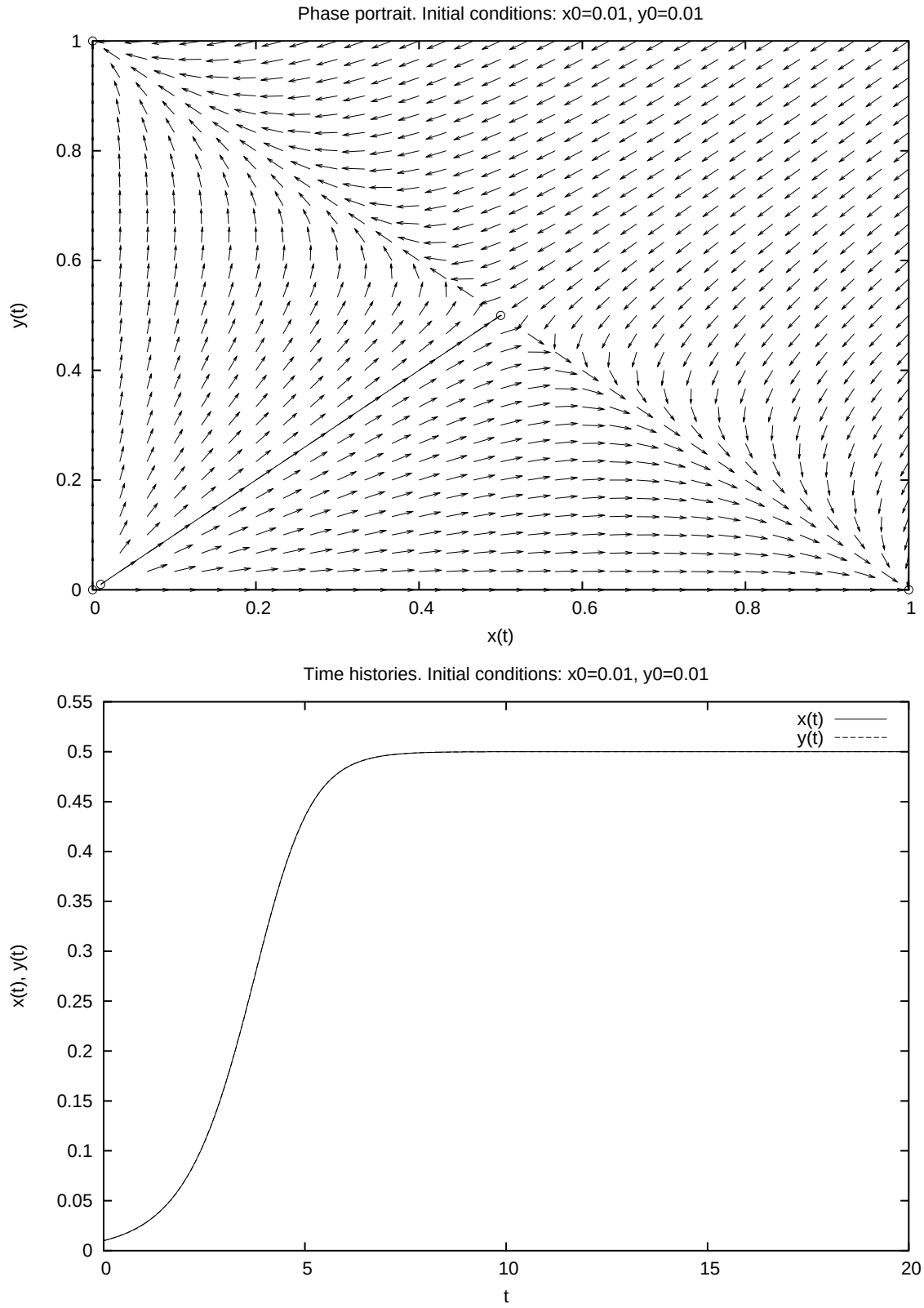


Figura 42: Ritratto di fase e storia temporale del sistema Lotka-Volterra modificato per $k = 2$, $(x_0, y_0) = (0.01, 0.01)$ e $t_f = 20$.

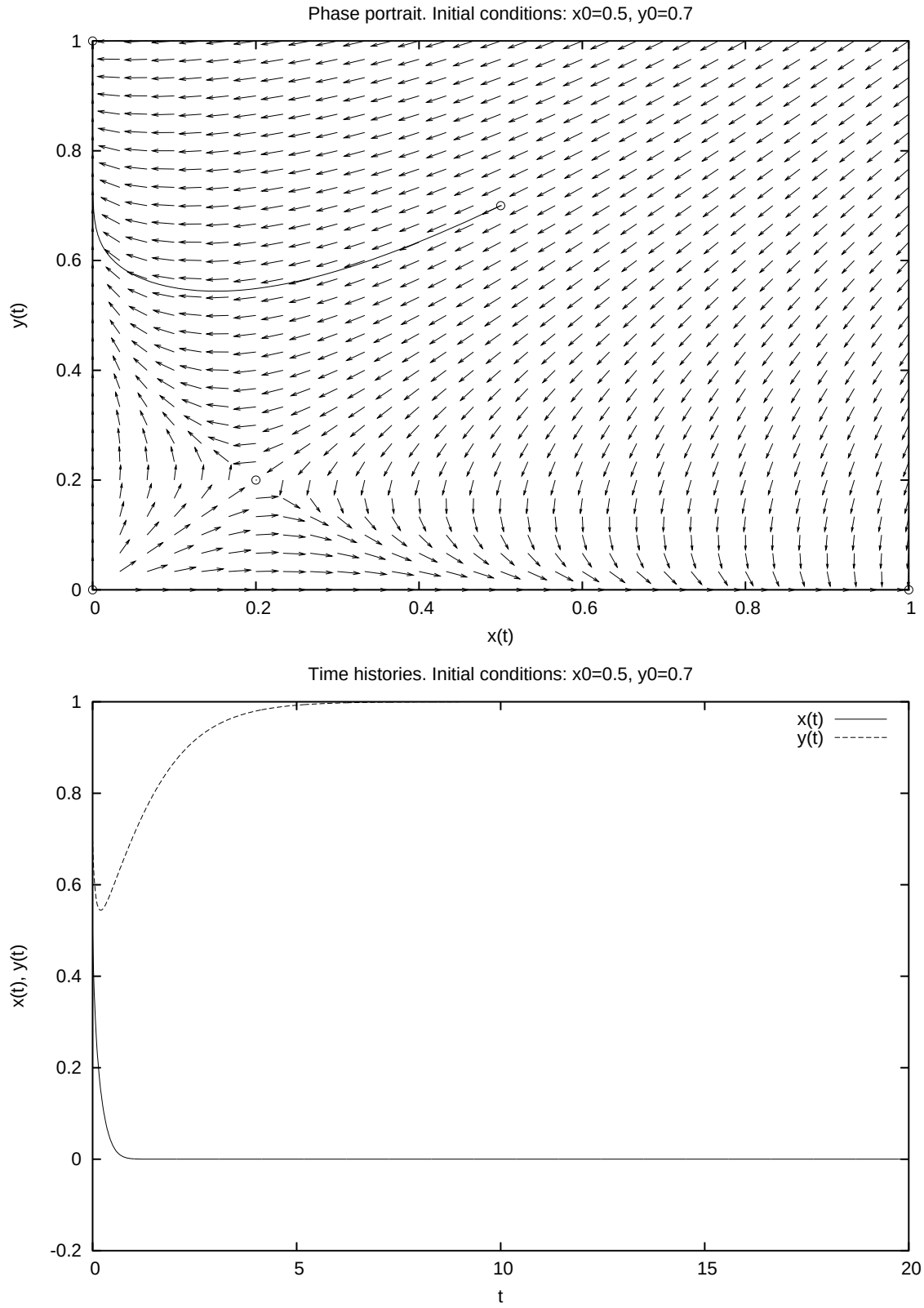


Figura 43: Ritratto di fase e storia temporale del sistema Lotka-Volterra modificato per $k = 20$, $(x_0, y_0) = (0.5, 0.7)$ e $t_f = 20$.

4.4 Esercizio

Studiare, al variare dei parametri $a, b \in \mathbb{R}, a \neq b$ (positivi e/o negativi), delle condizioni iniziali e del tempo finale, il comportamento del sistema (2).

4.5 Esercizio

Studiare, al variare delle condizioni iniziali e del tempo finale, il comportamento del sistema (4) per $0 < k \leq 1$.

4.6 Esercizio (Esclusione competitiva)

Studiare, al variare delle condizioni iniziali, dei coefficienti $\alpha, \beta, \gamma, \delta$ tutti positivi e tali che $0 < \delta << \beta$, e del tempo finale, il comportamento del sistema

$$\begin{cases} x' &= \alpha x - \beta x^2 - \gamma xy \\ y' &= \alpha y - (\beta - \delta) y^2 - \gamma xy. \end{cases}$$

4.6.1 Risoluzione

Il sistema dato è un caso particolare di un sistema (3) con $\alpha_1 = \alpha_1 = \alpha$, $a_{12} = a_{21} = -\gamma$, $a_{11} = -\beta$ e $a_{21} = -(\beta - \delta)$. Questo, in pratica, significa che le due popolazioni hanno leggi di crescita praticamente simili tranne che per il coefficiente intraspecifico, che varia leggermente. Pertanto, si può studiare utilizzando lo script `compcoop.m`, visto in precedenza.

In figura 44 viene riportato il caso $\alpha = 10, \beta = 0.1, \gamma = 0.099, \delta = 0.002$, $(x_0, y_0) = (40, 40)$. Come si può notare i punti di equilibrio sono tre, di cui uno solo è stabile e la soluzione si porta proprio su questo stato di equilibrio che prevede l'estinzione della popolazione x e la sopravvivenza della y. Questo succede perchè, pur essendo le due leggi di crescita molto simili, la popolazione x ha un coefficiente di competizione intraspecifico leggermente maggiore dell'altra. La figura 45 è stata ottenuta con gli stessi valori dei coefficienti ($\alpha = 10, \beta = 0.1, \gamma = 0.099, \delta = 0.002$), ma cambiando la condizione iniziale a $(x_0, y_0) = (95, 5)$ in modo da essere molto vicini al punto di equilibrio dove la specie x sopravvive e la specie y si estingue. Nonostante questo, la specie x si estingue e sopravvive solo la y. Si noti che, per questa scelta dei parametri, $\beta - \delta = 0.098 < \gamma$.

In figura 46 viene riportato il caso $\alpha = 10, \beta = 0.1, \gamma = 0.097, \delta = 0.002$, $(x_0, y_0) = (95, 5)$. Innanzitutto va sottolineato il fatto che i punti di equilibrio sono quattro, di cui quello stabile è il nuovo comparso. Questa volta la scelta dei coefficienti è tale per cui $\beta - \delta = 0.098 > \gamma$ ed il cambiamento sostanziale sta nel fatto che esiste un punto di equilibrio stabile che prevede la sopravvivenza di entrambe le specie. Questo succede partendo anche da altri dati iniziali, come mostrato in figura 47.

4.7 Esercizio

Si dimostri analiticamente come, al variare dei coefficienti $\alpha, \beta, \gamma, \delta$ tutti positivi e tali che $0 < \delta << \beta$, varia la natura dei punti di equilibrio del sistema

$$\begin{cases} x' &= \alpha x - \beta x^2 - \gamma xy \\ y' &= \alpha y - (\beta - \delta) y^2 - \gamma xy. \end{cases}$$

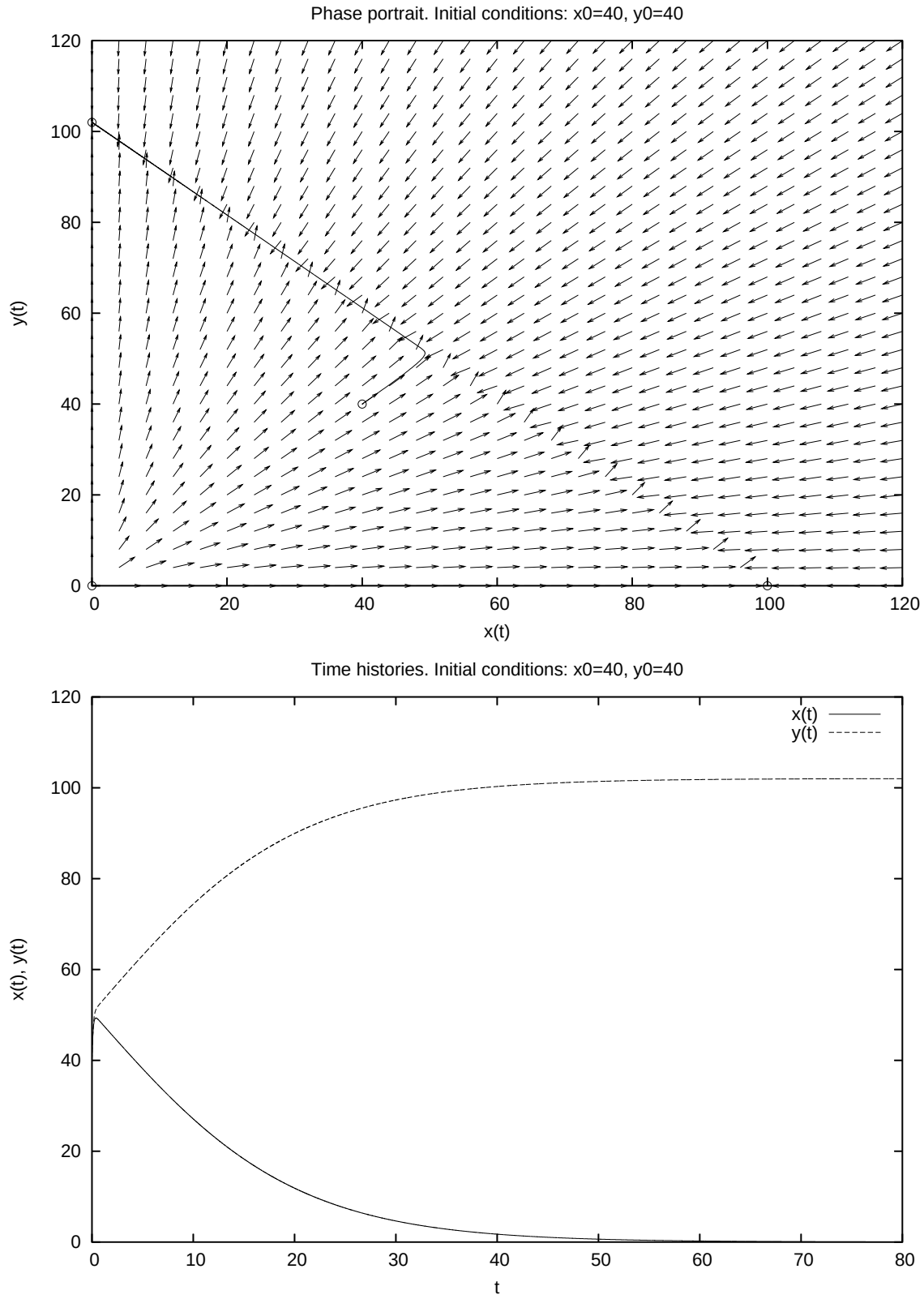


Figura 44: Ritratto di fase e storia temporale per $\alpha = 10, \beta = 0.1, \gamma = 0.099, \delta = 0.002$, $(x_0, y_0) = (40, 40)$ e $t_f = 80$.

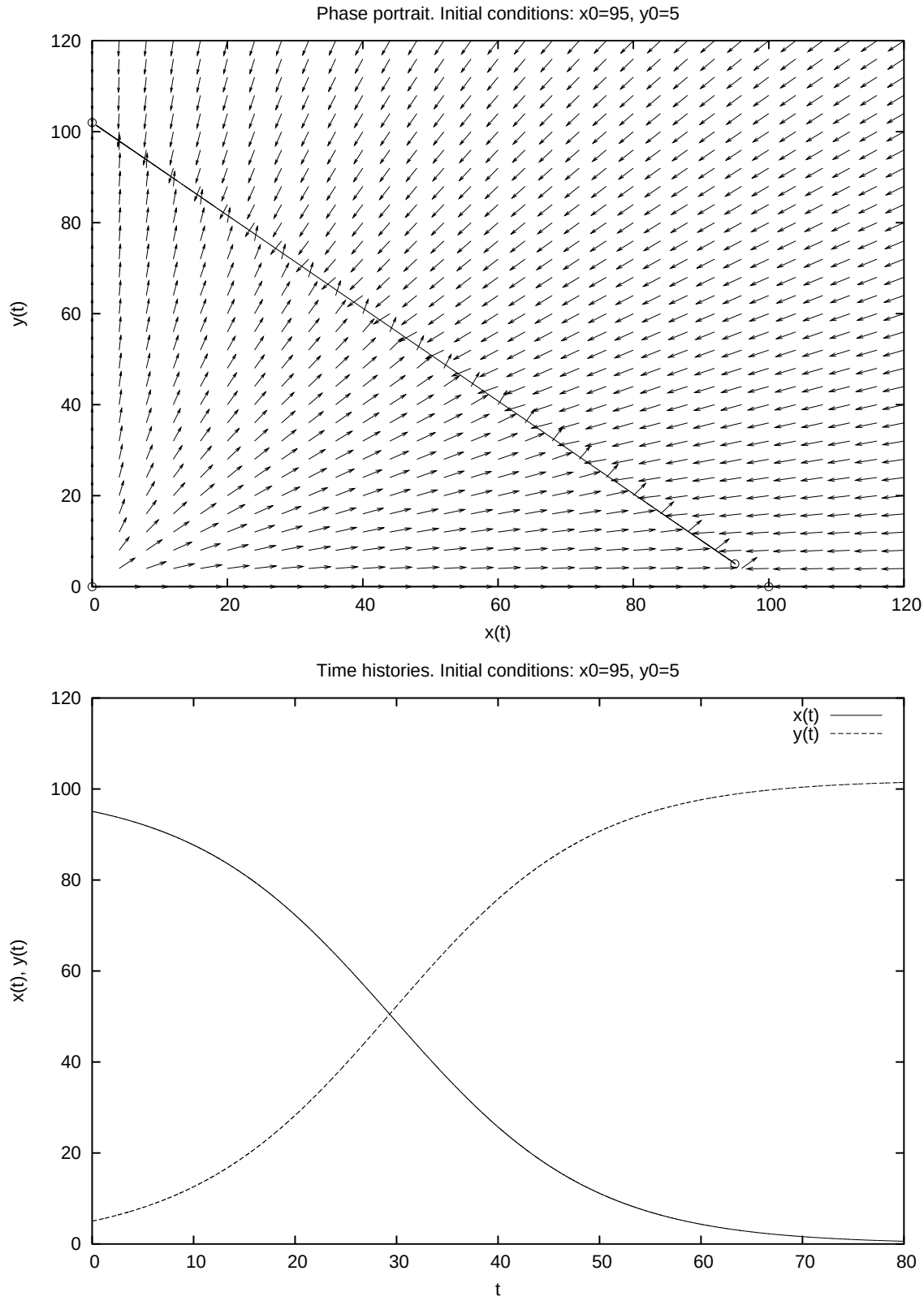


Figura 45: Ritratto di fase e storia temporale per $\alpha = 10, \beta = 0.1, \gamma = 0.099, \delta = 0.002$, $(x_0, y_0) = (95, 5)$ e $t_f = 80$.

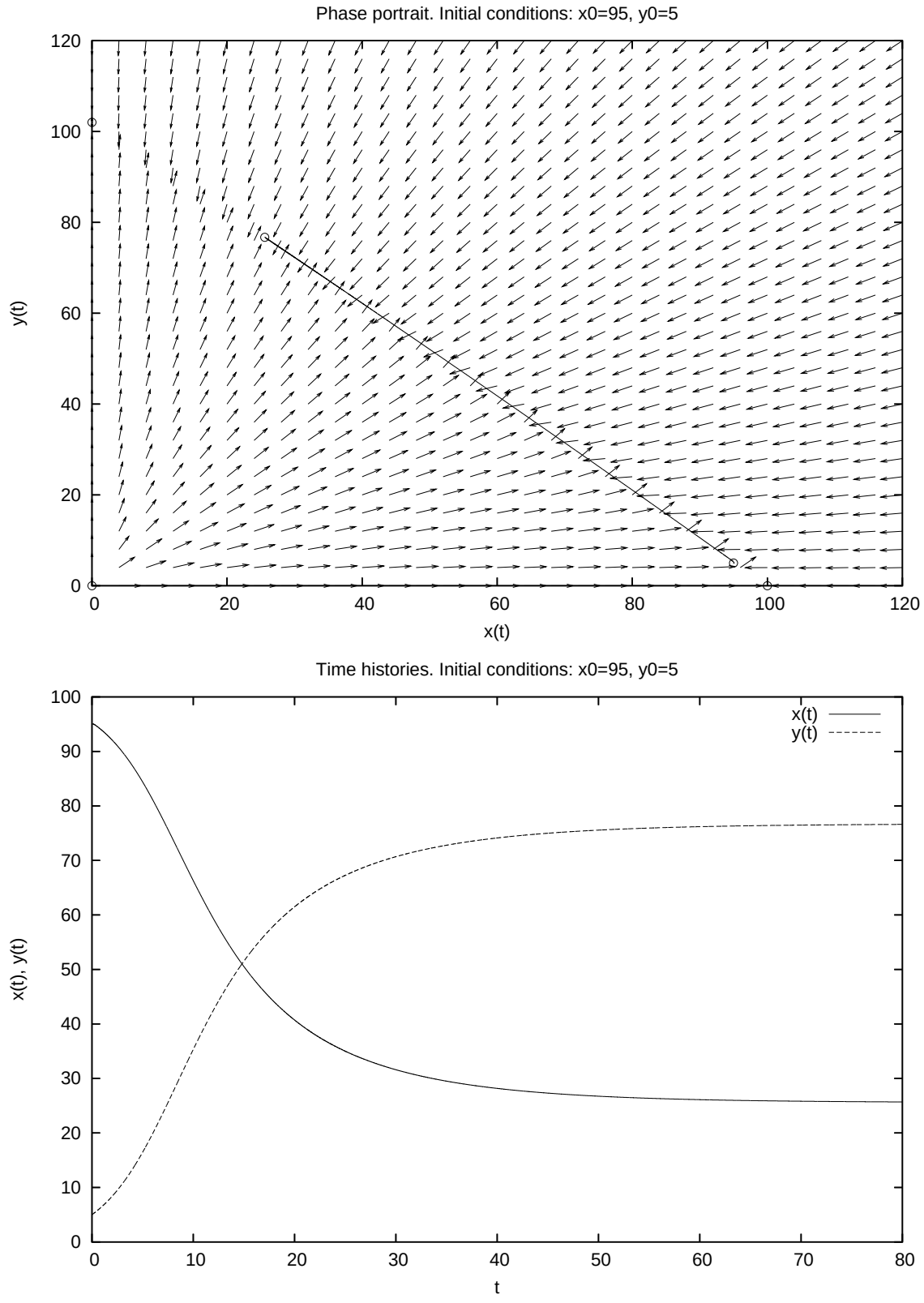


Figura 46: Ritratto di fase e storia temporale per $\alpha = 10, \beta = 0.1, \gamma = 0.097, \delta = 0.002$, $(x_0, y_0) = (95, 5)$ e $t_f = 80$.

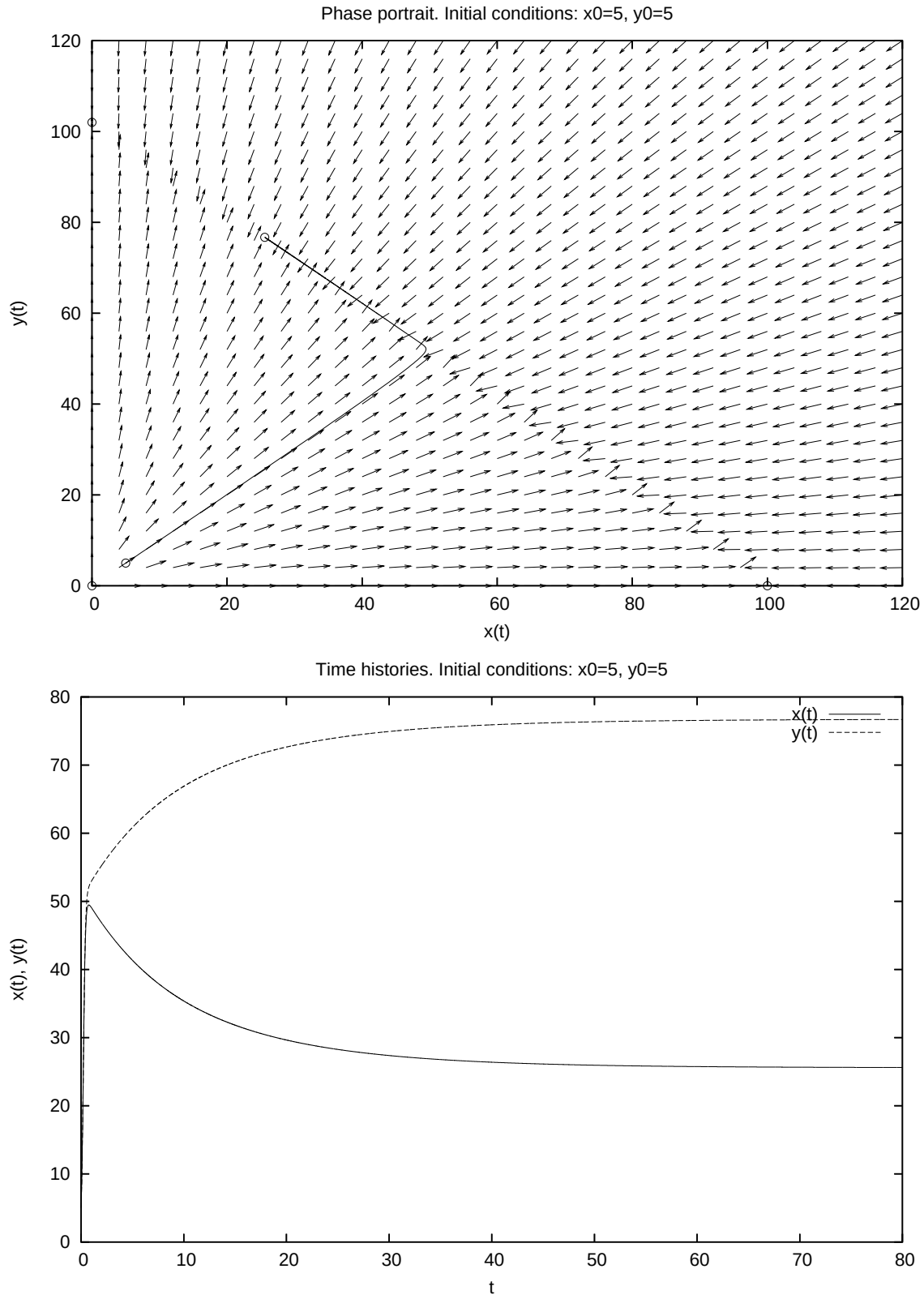


Figura 47: Ritratto di fase e storia temporale per $\alpha = 10, \beta = 0.1, \gamma = 0.097, \delta = 0.002$, $(x_0, y_0) = (5, 5)$ e $t_f = 80$.

5 Complessità dei sistemi competitivi (cicli limite, ampie oscillazioni, attrattori strani)

5.1 Esercizio

Studiare, al variare dei parametri a, b, c (positivi), delle condizioni iniziali e del tempo finale, il comportamento del sistema

$$\begin{cases} x' &= x(1-x) - \dfrac{axy}{x+c} \\ y' &= by(1-y/x). \end{cases}$$

5.1.1 Risoluzione

Il sistema ammette più punti di equilibrio, ma essendo le popolazioni x e y solo positive o nulle noi ci concentriamo esclusivamente in questo range ottenendo come unico punto di equilibrio $(\frac{(1-a-c)+\sqrt{(1-a-c)^2+4c}}{2}, \frac{(1-a-c)+\sqrt{(1-a-c)^2+4c}}{2})$, che indicheremo con $(x_{\text{eq}}, y_{\text{eq}})$. Analizzando il sistema linearizzato attorno a $(x_{\text{eq}}, y_{\text{eq}})$, si scopre che la parte reale degli autovalori λ è negativa solo se

$$b > \frac{(a-\sqrt{(1-a-c)^2+4c})(1+a+c-\sqrt{(1-a-c)^2+4c})}{2a}.$$

Pertanto, assumendo $a, c > 0$ e plottando la superficie $b = \frac{(a-\sqrt{(1-a-c)^2+4c})(1+a+c-\sqrt{(1-a-c)^2+4c})}{2a}$ si ottiene visivamente la regione dello spazio (a, c, b) stabile e instabile (vedi figura 48). Come si può notare dalla figura 48, per $a < 1/2$ e tutti i valori di $b > 0, c > 0$, la stabilità del punto di equilibrio $(x_{\text{eq}}, y_{\text{eq}})$ è assicurata. Al contrario quando $a > 1/2$, affinché ci sia stabilità il generico punto (a, c, b) deve trovarsi al di sopra della superficie riportata in figura 48. In presenza di autovalori reali negativi $(x_{\text{eq}}, y_{\text{eq}})$ è un nodo stabile, mentre diventa un fuoco stabile se gli autovalori sono complessi a parte reale negativa. Nel caso instabile si passa da nodo instabile (entrambi gli autovalori reali positivi) a fuoco instabile (autovalori complessi a parte reale positiva).

In figura 49 sono riportati i risultati ottenuti con $a = 0.1, b = 0.1, c = 0.1$, partendo dalla condizione iniziale $(x_0, y_0) = (0.1, 0.1)$. Come si può notare, il punto di equilibrio è stabile. Il caso $a = 0.1, b = 5, c = 1$, $(x_0, y_0) = (0.1, 0.1)$ (figura 50), ottenuto per valori dei parametri che assicurano la stabilità, mostra un chiaro fuoco stabile. Mantenendo fissa la condizione iniziale cambiando i paramentri in modo da avvicinarsi al limite di stabilità, si osserva che la traiettoria compie sempre più giri attorno al punto di equilibrio stabile, indice del fatto che la parte reale degli autovali, pur essendo negativa, è piccola rispetto alla parte immaginaria (si veda la figura 51, ottenuta con $a = 1, b = 0.5, c = 0.2$).

Nel caso in cui la condizione di stabilità non sia rispettata, il punto di equilibrio diventa instabile per cui, anche partendo da condizioni iniziali molto vicine ad esso, la traiettoria se ne allontana. Per l'analisi di questa situazione è possibile applicare il teorema di Poincaré-Bendixson che assicura l'esistenza di un ciclo limite cui tende la

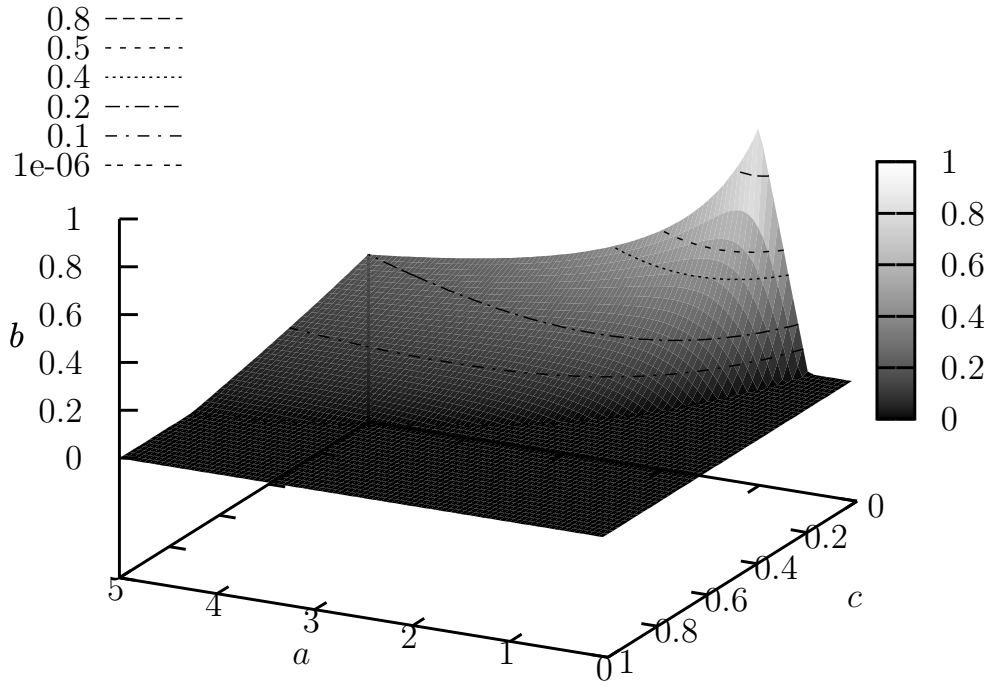


Figura 48: Il punto di equilibrio risulta stabile per le terne (a, c, b) che si trovano al di sopra della superficie riportata.

traiettoria quando si parta da condizioni iniziali interne ad esso. Quindi, partendo all'interno di questo ciclo limite la traiettoria si avvicinerà a tale ciclo e partendo all'esterno altrettanto.

In figura 52 è riportato il caso $a = 1, b = 0.1, c = 0.1$, che assicura l'instabilità del punto $(x_{\text{eq}}, y_{\text{eq}}) \approx (0.27016, 0.27016)$. Partendo da una condizione iniziale $(x_0, y_0) = (0.27, 0.27)$ molto vicina a $(x_{\text{eq}}, y_{\text{eq}})$, si può notare un'iniziale allontanamento dalla condizione iniziale e il susseguente carattere periodico della traiettoria quando raggiunge il ciclo limite stabile. La figura 53 differisce dalla 52 solo per le condizioni iniziali $(x_0, y_0) = (0.2, 0.01)$, che sono ora esterne al cilco limite che si ottiene per $a = 1, b = 0.1, c = 0.1$. Come si può notare, la traiettoria viene attratta dal ciclo limite che risulta, pertanto, stabile.

Si noti una differenza sostanziale tra il Lotka-Volterra classico visto la volta scorsa e il modello più realistico. Ora il ciclo limite è stabile, nel senso che attrae le traittorie. Al contrario, il ciclo limite del sistema classico è tale per cui una leggera variazione delle condizioni iniziali, ovvero una perturbazione nell'evoluzione ciclica, può portare a traiettorie che per alcuni tempi possono essere molto lontane (nel piano delle fasi). Questo si verifica, in particolare, quando la perturbazione avviene per valori delle variabili tali per cui il prodotto xy è piuttosto piccolo.

La figura 48 è stata ottenuta utilizzando lo script per GNU Gnuplot riportato in appendice A.

Utilizzando il file `realLV.m` riportato in appendice A si possono fare diverse prove al variare dei parametri a, b, d, della condizione iniziale e del tempo finale di integrazione. **Si ricordi di cambiare gli estremi della finestra di visualizzazione dipendentemente dalla condizione iniziale**.

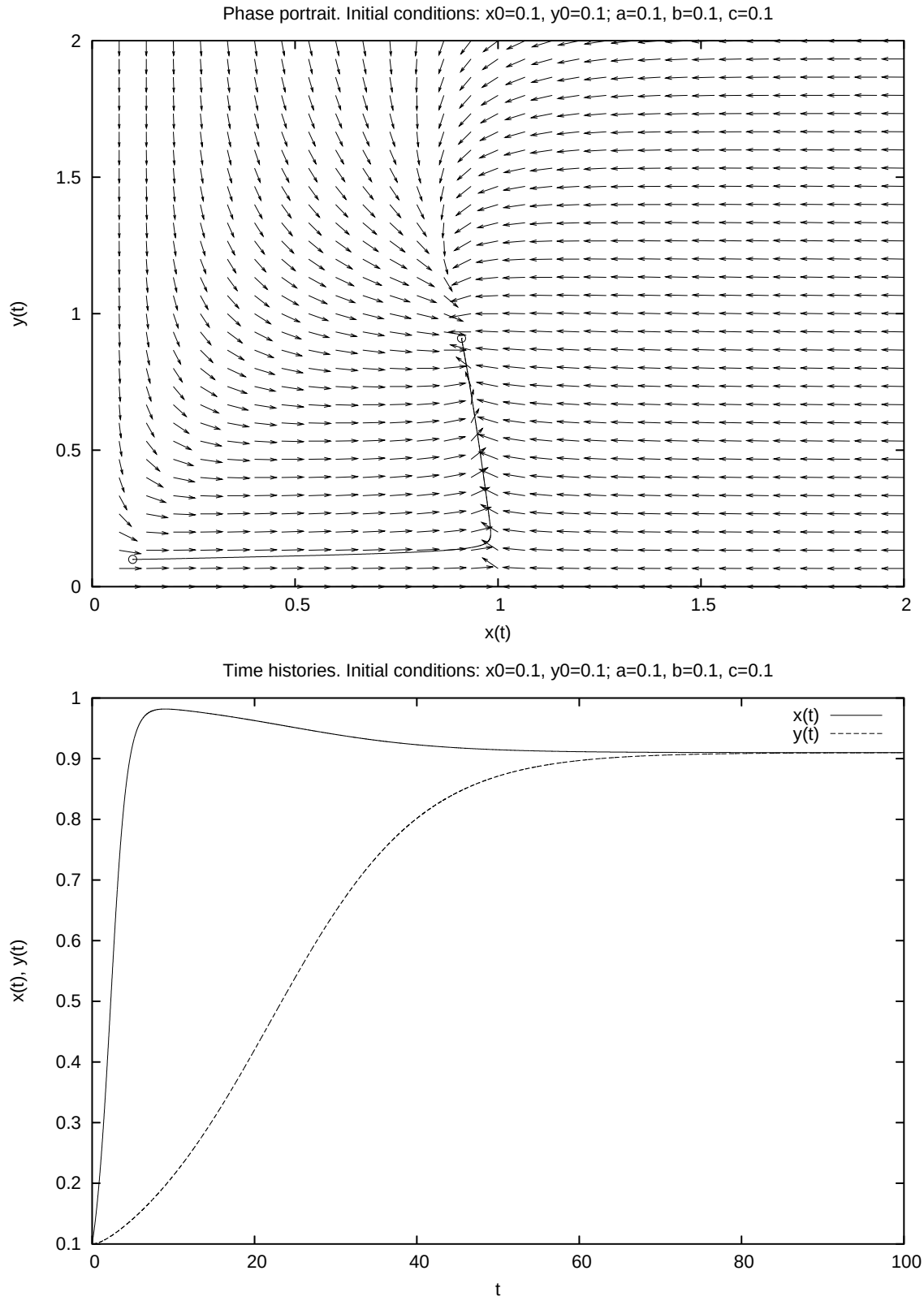


Figura 49: Ritratto di fase e storia temporale della versione realistica del modello Lotka-Volterra per $a = 0.1, b = 0.1, c = 0.1$, $(x_0, y_0) = (0.1, 0.1)$ e $t_f = 100$.

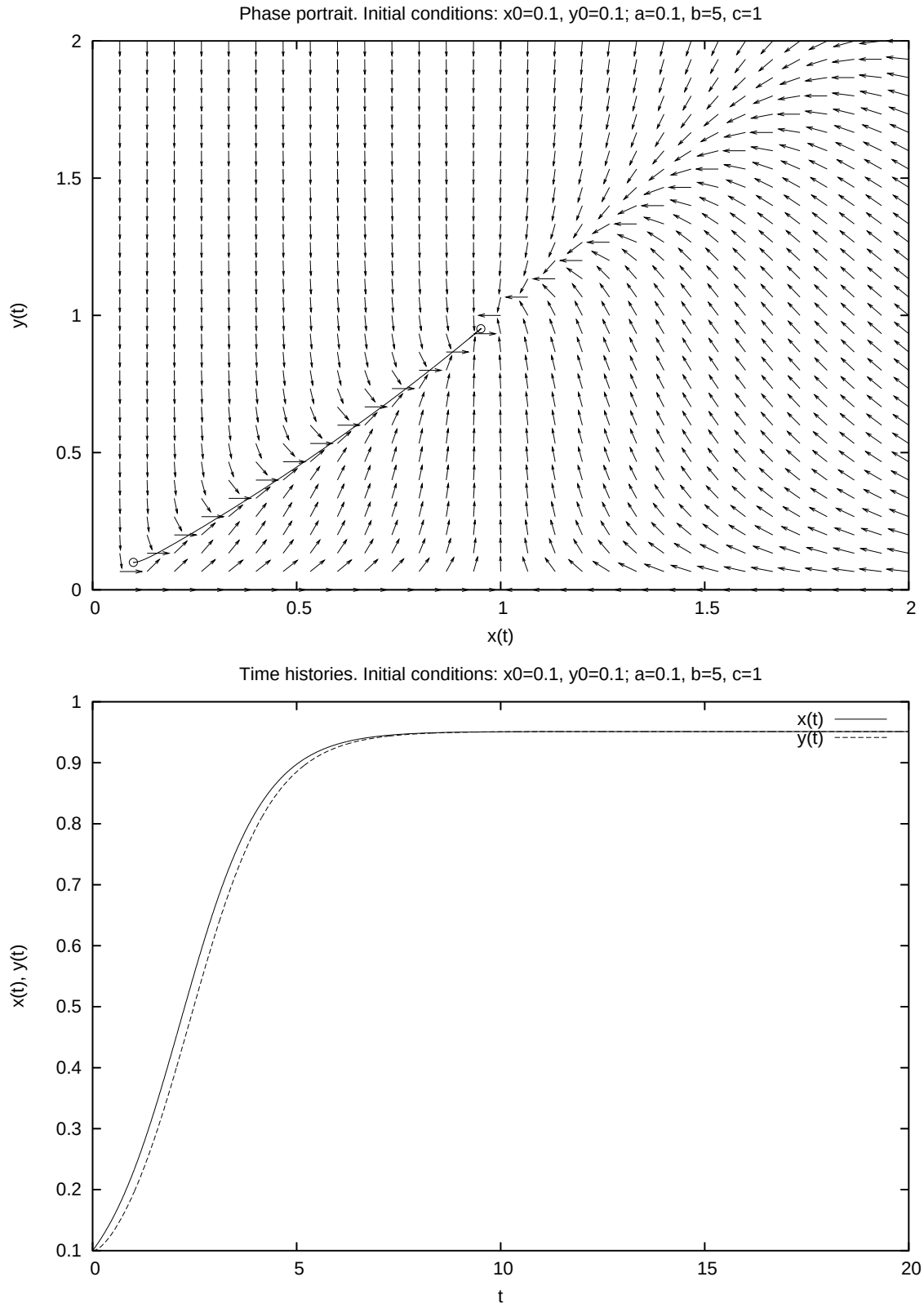


Figura 50: Ritratto di fase e storia temporale della versione realistica del modello Lotka-Volterra per $a = 0.1, b = 5, c = 1$, $(x_0, y_0) = (0.1, 0.1)$ e $t_f = 20$.

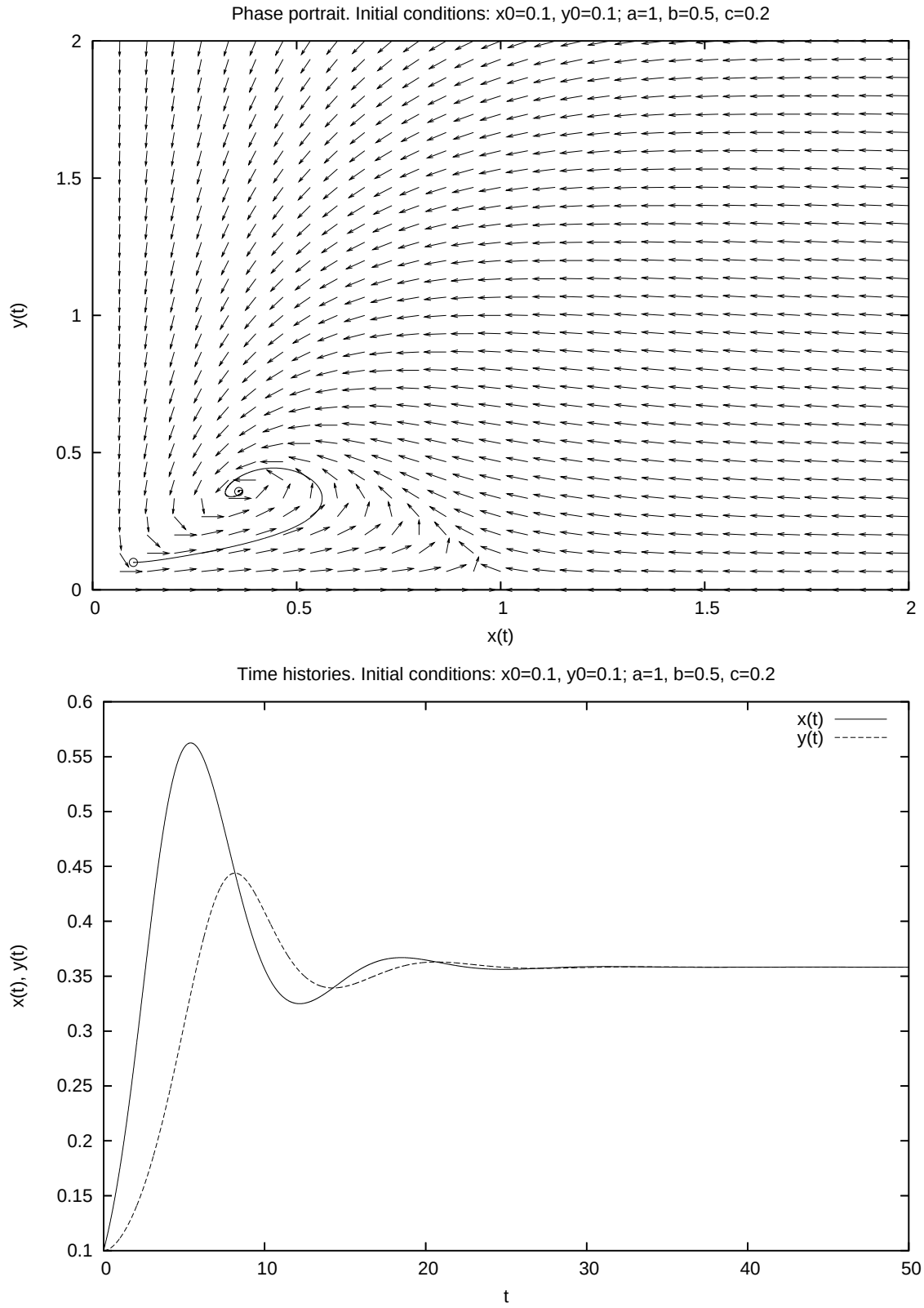


Figura 51: Ritratto di fase e storia temporale della versione realistica del modello Lotka-Volterra per $a = 1, b = 0.5, c = 0.2$, $(x_0, y_0) = (0.1, 0.1)$ e $t_f = 50$.

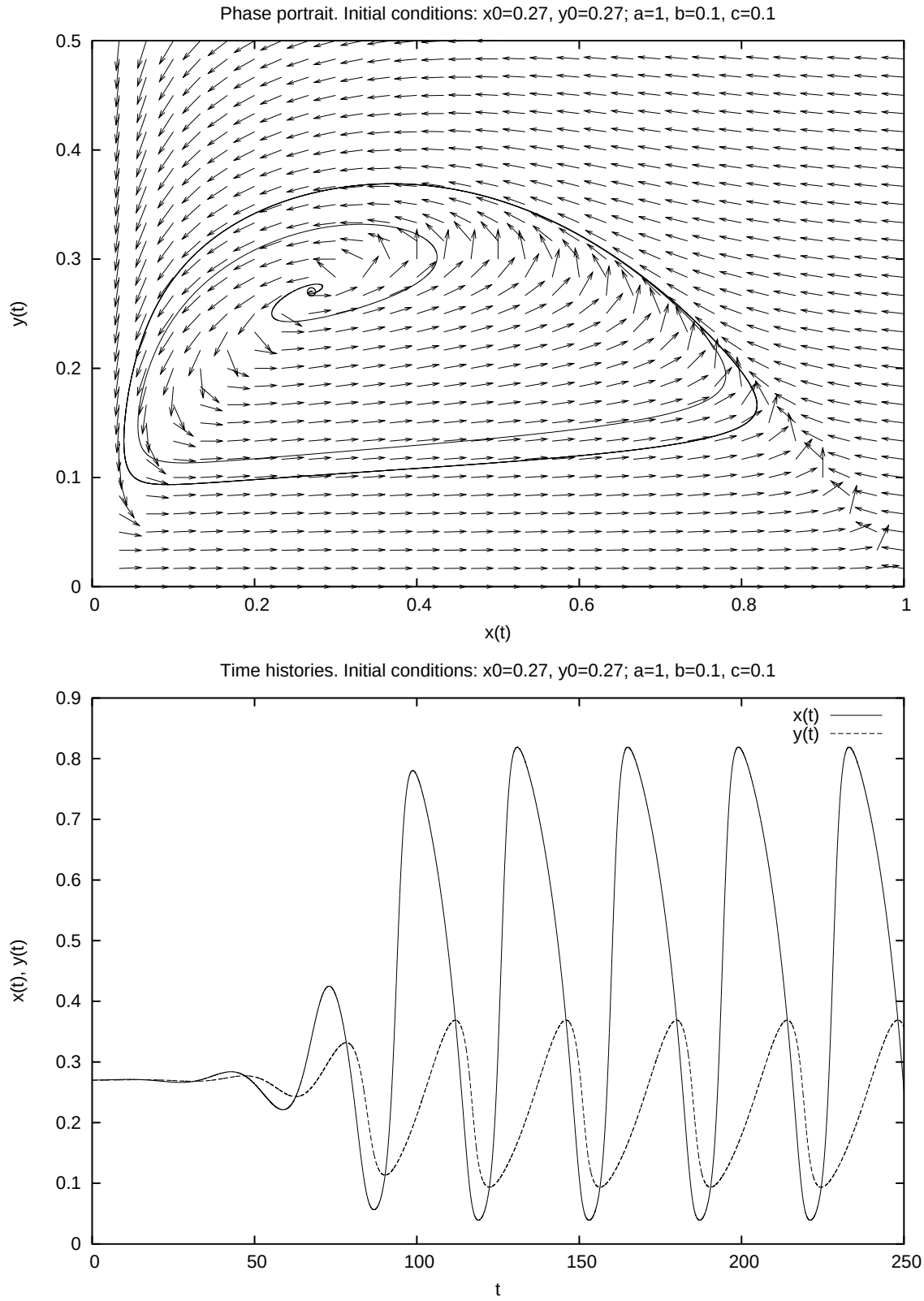


Figura 52: Ritratto di fase e storia temporale della versione realistica del modello Lotka-Volterra per $a = 1, b = 0.1, c = 0.1$, $(x_0, y_0) = (0.27, 0.27)$ e $t_f = 250$.

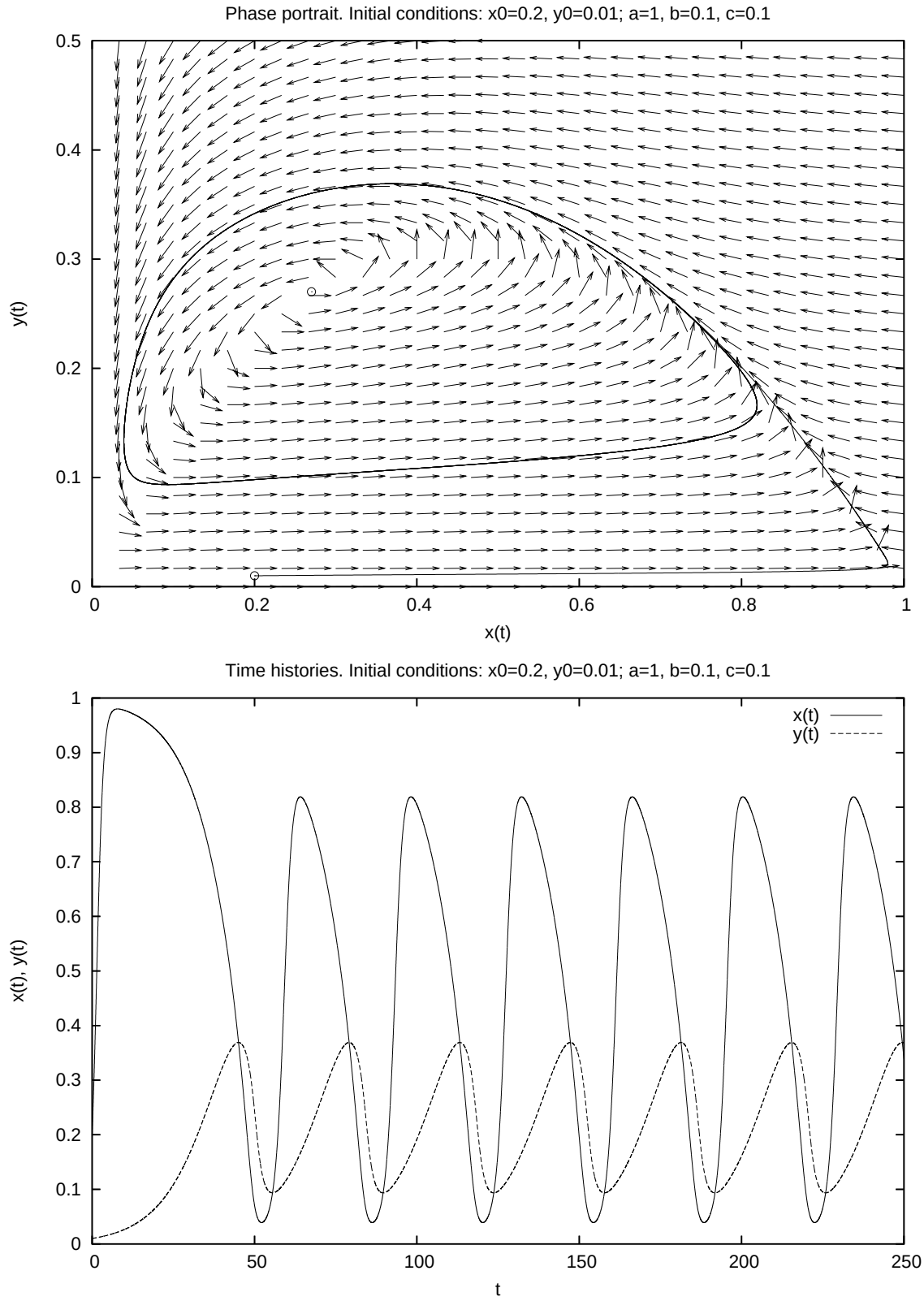


Figura 53: Ritratto di fase e storia temporale della versione realistica del modello Lotka-Volterra per $a = 1, b = 0.1, c = 0.1$, $(x_0, y_0) = (0.2, 0.01)$ e $t_f = 250$.

5.2 Esercizio

Studiare, al variare delle condizioni iniziali e del tempo finale, il comportamento del sistema

$$\begin{cases} x' &= x\left[\left(1+\dfrac{1}{1+(x-2)^2}\right)-y\right] \\ y' &= y\left[x-(y+1)\right]. \end{cases}$$

5.2.1 Risoluzione

Il sistema dato è un caso particolare di sistemi del tipo

$$\begin{cases} x' &= x\left[F(x)-y\right]=f(x,y) \\ y' &= y\left[x-G(y)\right]=g(x,y), \end{cases}$$

con $F(x) = 1 + \dfrac{1}{1+(x-2)^2}$ e $G(y) = y + 1$, che ammette come punti di equilibrio l'origine e $(x_0, y_0) \approx (2.6823, 1.6823)$. Si può dimostrare che l'origine è un punto di sella (e quindi instabile) in quanto $\lambda_1 = F(0) > 0$ e $\lambda_2 = -G(0) < 0$, mentre l'altro punto di equilibrio, se è come in figura 54, risulta essere stabile (a piccole perturbazioni) qualsiasi siano $F(x)$ e $G(y)$ essendo la parte reale degli autovalori negativa.

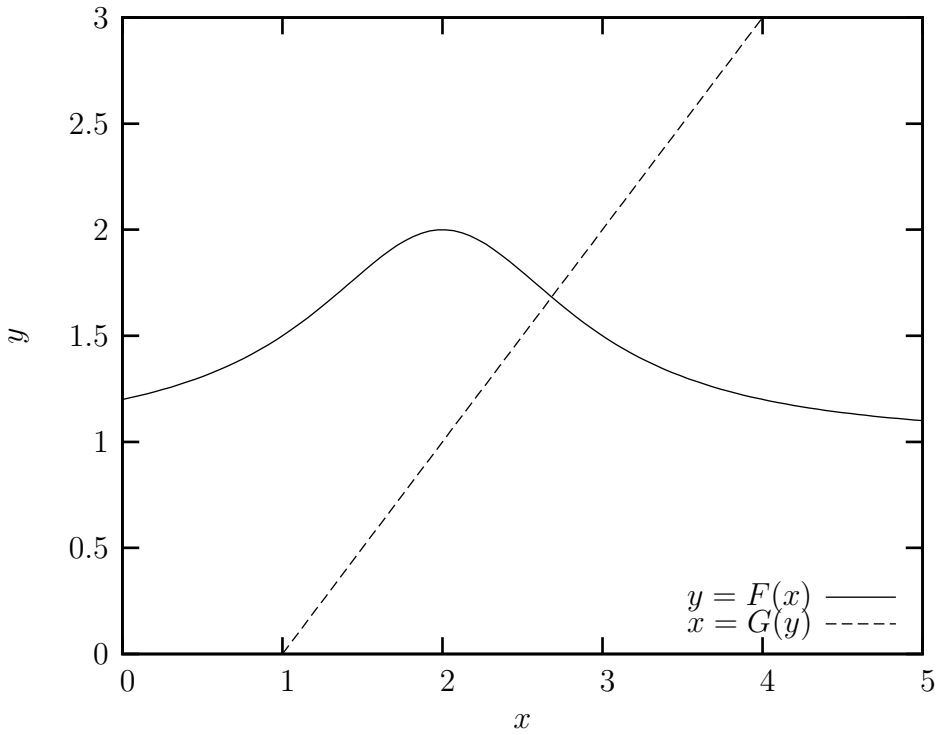


Figura 54: Luoghi dei punti $y = F(x)$ e $x = G(y)$, ovvero $f(x, y) = 0$ e $g(x, y) = 0$.

Pertanto, perturbando l'equilibrio, ovvero partendo da una condizione iniziale sufficientemente vicina al punto di equilibrio, le traiettorie ne vengono attratte e compiono orbite piuttosto ristrette riportandosi rapidamente sulla soluzione di equilibrio (vedi figura 55).

Al contrario, dipendentemente da quanto è lontana la condizione iniziale dal punto di equilibrio, si possono ottenere traiettorie con ampie oscillazioni (vedi figure 56–58).

La condizione che discrimina tra l'uno e l'altro caso, come si può facilmente intuire dalla figura 54, è dove la traiettoria interseca la linea $f(x, y) = 0$. Se questo avviene per $x < \bar{x}$, dove $\bar{x}$ è la x dove $y = F(x)$ raggiunge il massimo, allora si hanno ampie oscillazioni, altrimenti no.

Molto del comportamento di questo sistema può essere dedotto dall'analisi delle isocline nulle, ovvero dei luoghi geometrici dei punti del piano delle fasi per i quali si ha $x' = 0$ (isocline verticali) e $y' = 0$ (isocline orizzontali). In pratica, questi luoghi sono proprio le curve riportate in figura 54, che dividono il piano in 4 parti che indicheremo con NE (nord-est), NO (nord-ovest), SO (sud-ovest) e SE (sud-est). Studiando il segno di $f(x, y)$ e $g(x, y)$ si deduce facilmente che la regione NE è caratterizzata da $x' < 0$ e $y' > 0$, quella NO da $x' < 0$ e $y' < 0$, quella SO da $x' > 0$ e $y' < 0$ e quella SE da $x' > 0$ e $y' > 0$. Si può quindi concludere che le orbite si avvolgono attorno al punto critico seguendo il verso antiorario. Si noti che, dalla sola analisi delle isocline, non si può dedurre la stabilità o meno del punto di equilibrio, ovvero se le orbite si avvicino o si allontanino da esso.

Utilizzando il file `tp.m` riportato in appendice A si possono fare diverse prove al variare della condizione iniziale e del tempo finale di integrazione. **Si ricordi di cambiare gli estremi della finestra di visualizzazione dipendentemente dalla condizione iniziale**.

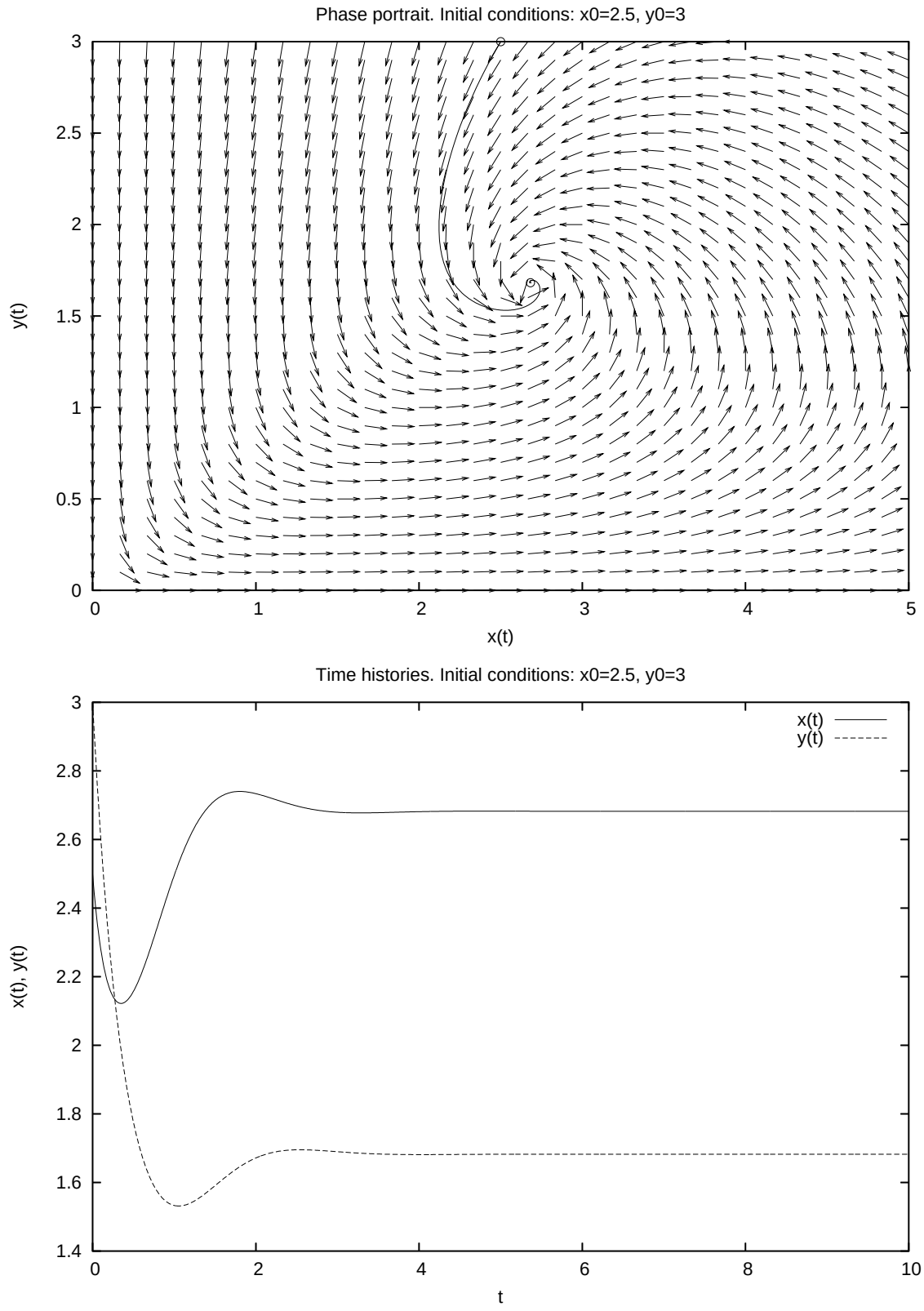


Figura 55: Ritratto di fase e storia temporale per $(x_0, y_0) = (2.5, 3.0)$ e $t_f = 10$.

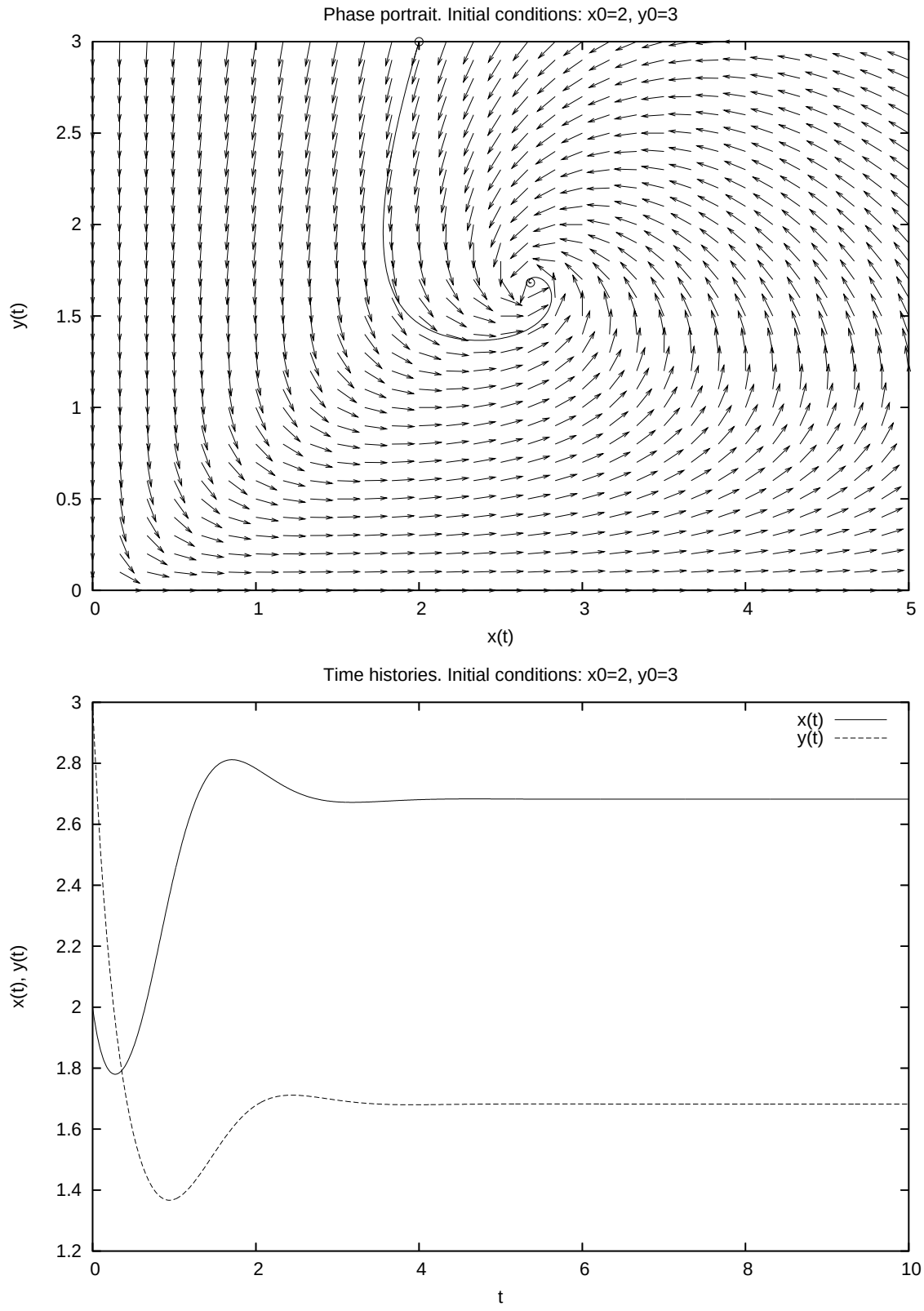


Figura 56: Ritratto di fase e storia temporale per $(x_0, y_0) = (2.0, 3.0)$ e $t_f = 10$.

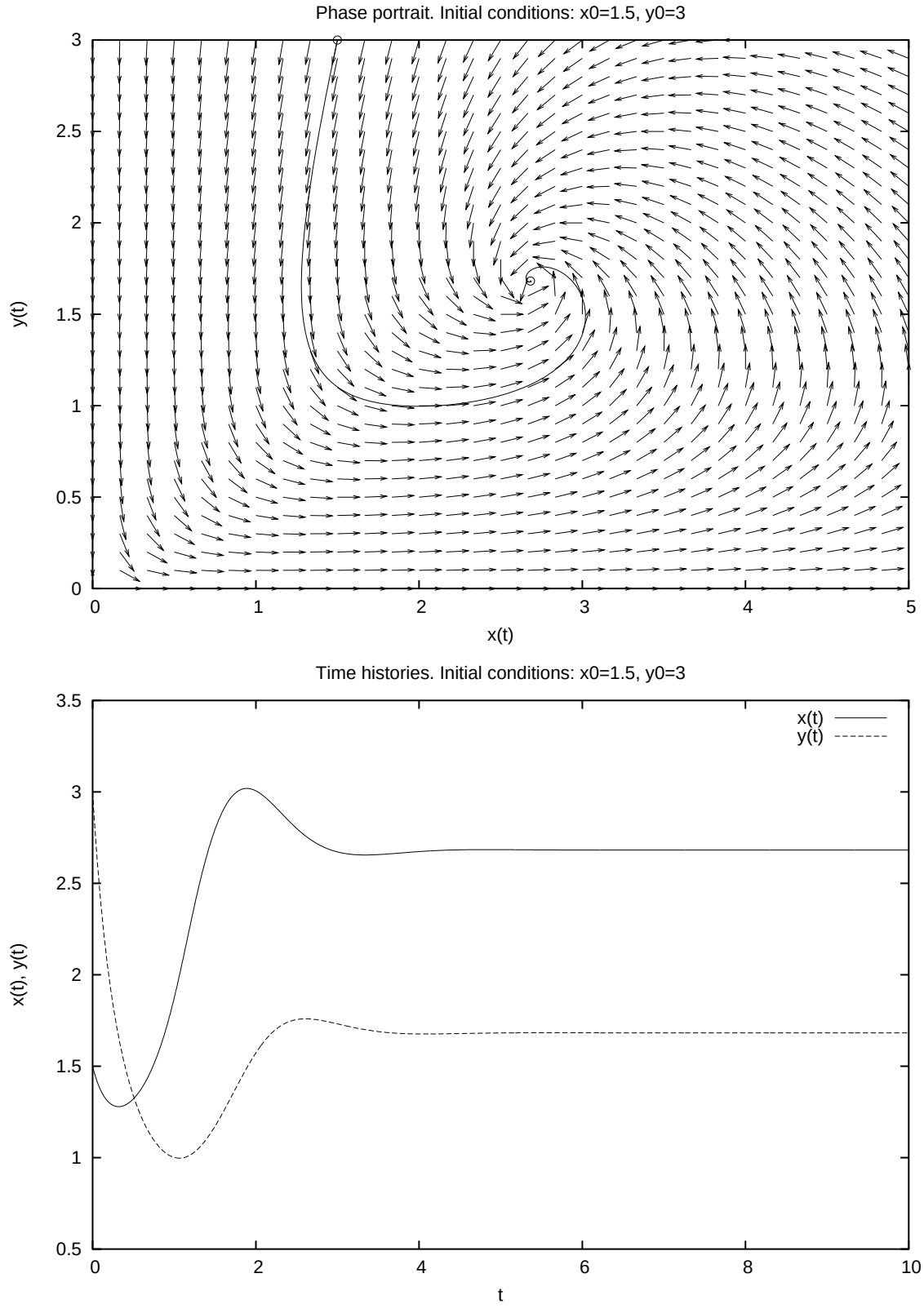


Figura 57: Ritratto di fase e storia temporale per $(x_0, y_0) = (1.5, 3.0)$ e $t_f = 10$.

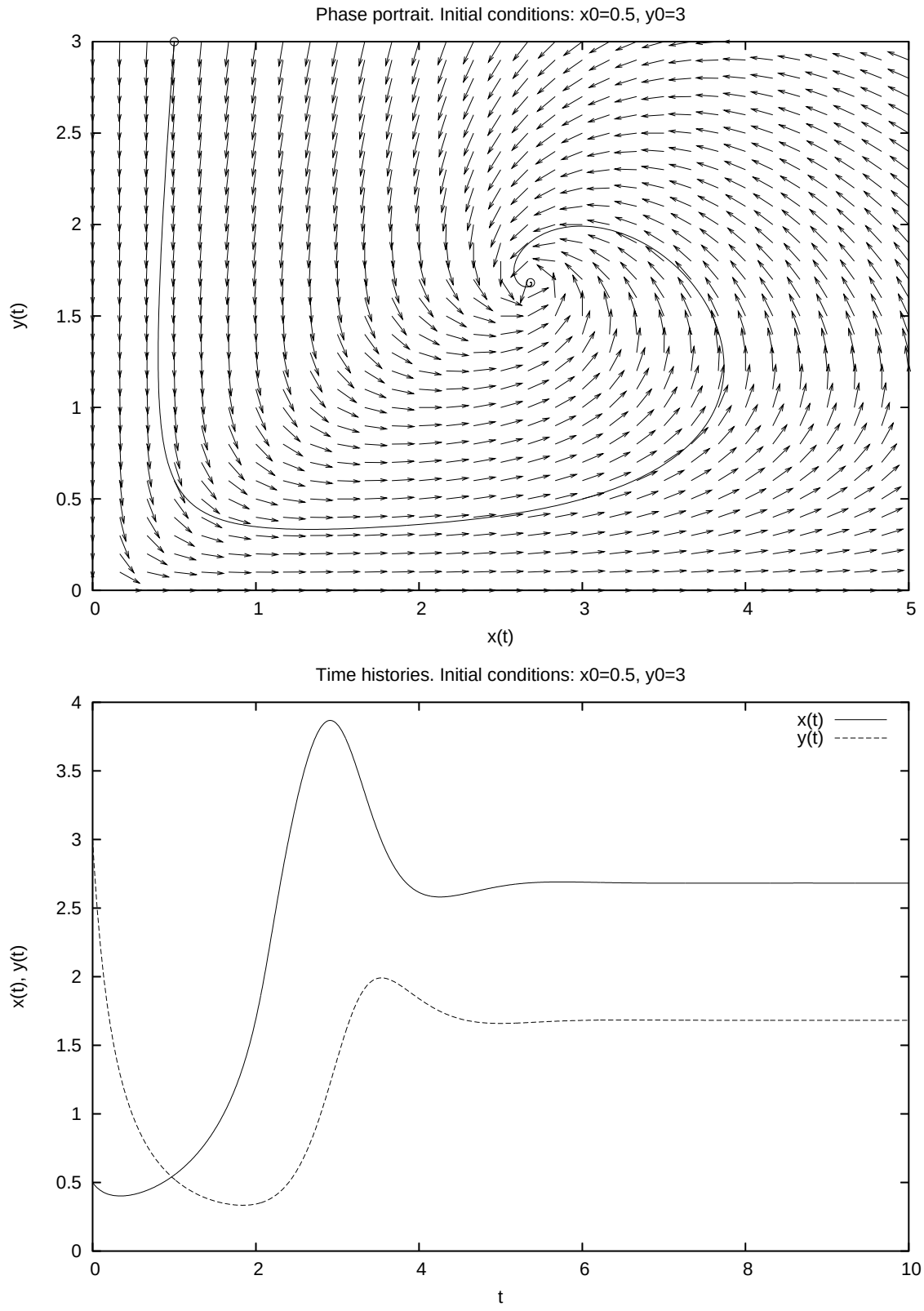


Figura 58: Ritratto di fase e storia temporale per $(x_0, y_0) = (0.5, 3.0)$ e $t_f = 10$.

5.3 Esercizio

Studiare, al variare delle condizioni iniziali e del tempo finale, il comportamento del sistema

$$\begin{cases} x' &= a(y-x) \\ y' &= -xz + bx - y \\ z' &= xy - cw. \end{cases}$$

5.3.1 Risoluzione

Il sistema in questione è il ben noto sistema di Lorenz inizialmente utilizzato in campo metereologico. Esso ammette tre punti di equilibrio, di cui uno è l'origine e gli altri sono simmetrici rispetto all'asse z e sono $(x, y, x) = (\sqrt{bc-c}, \sqrt{(b-1)c}, b-1)$ e $(x, y, x) = (-\sqrt{bc-c}, -\sqrt{(b-1)c}, b-1)$. Essi sono tutti instabili, con uno o due autovalori a parte reale positiva, come si può notare in figura 59.

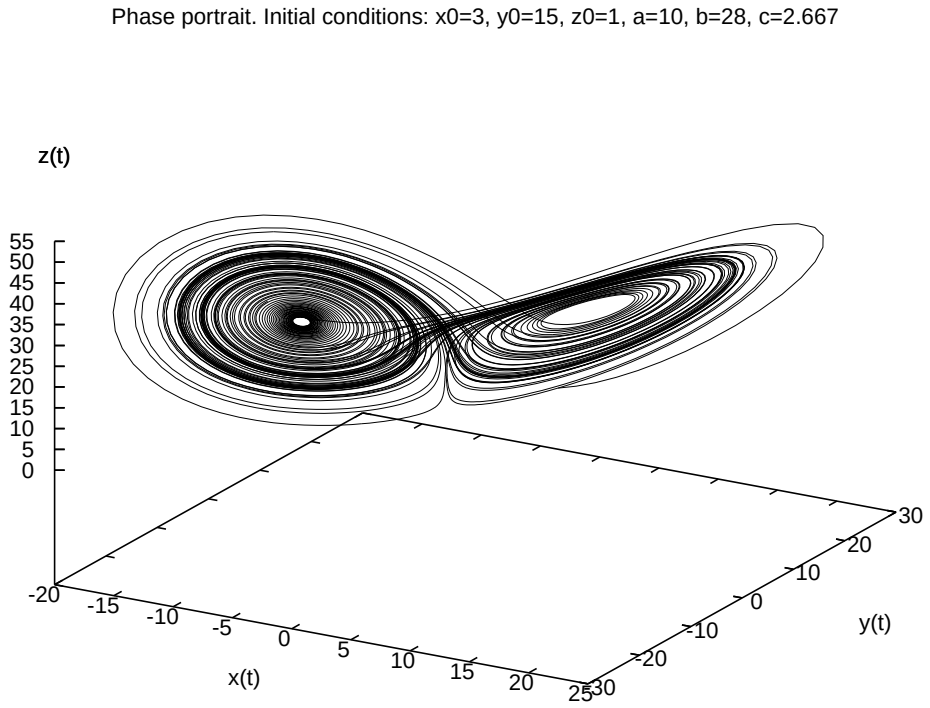


Figura 59: Attrattore di Lorenz ottenuto con $a = 10, b = 28, c = 8/3$, $(x_0, y_0, z_0) = (3151)$ e $t_f = 100$.

Utilizzando il file `mylorenz.m` riportato in appendice A si possono fare diverse prove al variare delle costanti a, b, c, della condizione iniziale e del tempo finale di integrazione.

A Scripts per GNU Octave

```
% Name:     esempio1.m
% Author:   Simone Zuccher
% Created:  03 Apr 2007
% Purpose:  Compute x(k+1) = x(k) + sin(x(k))
% Input:    Number of total iterations and x(1)
% Output:   Plot of x(k) versus k and x(k+1) versus x(k)
% Modified:

% Clear all variables
clear

% Change format to long exponential
format long e

% Input the number of total iterations
m=input('Input N (number of iterations): ');

% Input the initial value. If negative, it will be a random number
s(1)=input('Input 0<s(1)<pi (if <0 or >pi then random): ');

% Set s(1) random if out of range
if((s(1)>pi) || (s(1)<0))
  s(1)=pi*rand(1);
endif

% Display value of s(1)
disp(s(1));

% Assign x and y needed for 2nd plot
x(1)=s(1);
y(1)=0.0;

% Loop on all points
for n=1:1:m-1
  s(n+1)     = s(n)+sin(s(n));
  disp(s(n+1));
  x(2*n)     = s(n);
  y(2*n)     = s(n+1);
  x(2*n+1)   = s(n+1);
  y(2*n+1)   = s(n+1);
end

% Plots s(n) versus n
gset auto
plot(s,'+-g;s(n);');

% Wait for keypressed
disp('Please press a key to continue...');
pause();

% Plot f(x), x and path
t=linspace(0,pi,500);
plot(t,t+sin(t),'-g;x+sin(x);',t,t,'-b;x;',x,y,'+-');
```

```
% Name:     esempio2.m
% Author:   Simone Zuccher
% Created:  03 Apr 2007
% Purpose:  Compute x(k+1) = max(1/4,x(k)^2)
% Input:    Number of total iterations and x(1)
% Output:   Plot of x(k) versus k and x(k+1) versus x(k)
% Modified:

% Clear all variables
clear

% Change format to long exponential
format long e

% Input the number of total iterations
m=input('Input N (number of iterations): ');

% Input the initial value.
s(1)=input('Input s(1): ');

% Display value of s(1)
disp(s(1));

% Assign x and y needed for 2nd plot
x(1)=s(1);
y(1)=0.0;

% Loop on all points
for n=1:1:m-1
  s(n+1)     = max(1/4,s(n)^2);
  disp(s(n+1));
  x(2*n)     = s(n);
  y(2*n)     = s(n+1);
  x(2*n+1)   = s(n+1);
  y(2*n+1)   = s(n+1);
end

% Plots s(n) versus n
gset auto
plot(s,'+-g;s(n);');

% Wait for keypressed
disp('Please press a key to continue...');
pause();

% Display value of s(m)
disp(s(m));

% Plot f(x), x and path
t=linspace(-1.5,1.5,500);
plot(t,max(1/4,t.^2),'-g;max(1/4,t^2);',t,t,'-b;x;',x,y,'+-');
```

```
% Name:     esempio3.m
% Author:   Simone Zuccher
% Created:  03 Apr 2007
% Purpose:  Compute x(k+1) = integrate(exp(-t^2),t,0,x(k))
% Input:    Number of total iterations and x(1)
% Output:   Plot of x(k) versus k and x(k+1) versus x(k)
% Modified:

% Clear all variables
clear

% Change format to long exponential
format long e

% Input the number of total iterations
m=input('Input N (number of iterations): ');

% Input the initial value.
s(1)=1.0;

% Display value of s(1)
disp(s(1));

% Assign x and y needed for 2nd plot
x(1)=s(1);
y(1)=0.0;

% Loop on all points
for n=1:1:m-1
  s(n+1)     = sqrt(pi)*erf(s(n))/2.0;
  disp(s(n+1));
  x(2*n)     = s(n);
  y(2*n)     = s(n+1);
  x(2*n+1)   = s(n+1);
  y(2*n+1)   = s(n+1);
end

% Plots s(n) versus n
gset auto
plot(s,'+-g;s(n);');

% Wait for keypressed
disp('Please press a key to continue...');
pause();

disp(s(m))

% Plot f(x), x and path
t=linspace(0,1.5,500);
plot(t,sqrt(pi)*erf(t)/2.0,'-g;sqrt(pi)*erf(x)/2;',t,t,'-b;x;',x,y,'+-');
```

```
% Name:     esempio4.m
% Author:   Simone Zuccher
% Created:  04 Apr 2007
% Purpose:  Compute the solution of x^k = cos (x/k)
% Input:    Number of total iterations and x0 for nonlinear solver
% Output:   Plot  x^k and cos (x/k)
% Modified:

% Clear all variables
clear

% Change format to long exponential
format long e

% Input the number of total iterations
m=input('Input N (number of iterations): ');

% Input the initial value for nonlinear solver.
x0=input('Input x0 (initial value for nonlinear solver): ');

% Create a vector needed for plots
t=linspace(-1.1,1.1,500);

% Loop on all functions
for n=1:1:m
  % set range for plot
  gset xrange[-1.1:1.1]

  % set attributes for plots
  attr1=['-g;t^' int2str(n) ';'];
  attr2=['-b;cos(t/' int2str(n) ');'];

  % plot of x^n and cos(x/n)
  plot(t,t.^n,attr1,t,cos(t/n),attr2);

  % define the function
  fun= ["x^" int2str(n) "-cos(x/" int2str(n) ")"];

  % compute zero closest to x0 and diplay it
  s(n)=fsolve(fun,x0);
  disp(s(n))

  % Wait for keypressed
  disp('Please press a key to continue...');
  pause();
end
```

```
% Name:     esempio5.m
% Author:   Simone Zuccher
% Created:  04 Apr 2007
% Purpose:  Compute x(k+1) = x(k)/( 1 + x(k) )
% Input:    Number of total iterations and x(1)
% Output:   Plot of x(k) versus k and x(k+1) versus x(k)
% Modified:

% Clear all variables
clear

% Change format to long exponential
format long e

% Input the number of total iterations
m=input('Input N (number of iterations): ');

% Input the initial value.
s(1)=input('Input s(1) (initial value): ');

if(s(1)<0)
  disp('s(1)<0: stopping...')
  return
endif

% Display value of s(1)
disp(s(1));

% Assign x and y needed for 2nd plot
x(1)=s(1);
y(1)=0.0;

% Loop on all points
for n=1:1:m-1
  s(n+1)     = s(n)/(1.+s(n));
  disp(s(n+1));
  x(2*n)     = s(n);
  y(2*n)     = s(n+1);
  x(2*n+1)   = s(n+1);
  y(2*n+1)   = s(n+1);
end

% Plots s(n) versus n
gset auto
plot(s,'+-g;s(n);');

% Wait for keypressed
disp('Please press a key to continue...');
pause();

disp(s(m))

% Plot f(x), x and path
t=linspace(0,1,500);
plot(t,t./(1.+t),'-g;t/(1+t);',t,t,'-b;x;',x,y,'+-');
```

```
% Name:     esempio6.m
% Author:   Simone Zuccher
% Created:  04 Apr 2007
% Purpose:  Compute x(k+1) = 4*integrate(4*(exp(2*t))/(1+exp(2*t))^2,t,o,x(k))
% Input:    Number of total iterations and x(1)
% Output:   Plot of x(k) versus k and x(k+1) versus x(k)
% Modified:

% Clear all variables
clear

% Change format to long exponential
format long e

% Input the number of total iterations
m=input('Input N (number of iterations): ');

% Input the initial value.
s(1)=input('Input s(1) (initial value): ');

% Display value of s(1)
disp(s(1));

% Assign x and y needed for 2nd plot
x(1)=s(1);
y(1)=0.0;

% Loop on all points
for n=1:1:m-1
  s(n+1)     = 1. - 2./(exp(2.*s(n))+1.);
  disp(s(n+1));
  x(2*n)     = s(n);
  y(2*n)     = s(n+1);
  x(2*n+1)   = s(n+1);
  y(2*n+1)   = s(n+1);
end

% Plots s(n) versus n
gset auto
plot(s,'+-g;s(n);');

% Wait for keypressed
disp('Please press a key to continue...');
pause();

disp(s(m))

% Plot f(x), x and path
t=linspace(-2,2,500);
plot(t,1. - 2./(exp(2.*t)+1.),'-g;1-2/(exp(2*t)+1);',t,t,'-b;x;',x,y,'+-');
```

```
% Name:     esempio7.m
% Author:   Simone Zuccher
% Created:  03 Apr 2007
% Purpose:  Compute x(k+1) = x(k) + [x(k-1)]^2
% Input:    Number of total iterations and x(2)
% Output:   Plot of x(k) versus k and x(k+1) versus x(k)
% Modified:

% Clear all variables
clear

% Change format to long exponential
format long e

% Input the number of total iterations
m=input('Input N (number of iterations): ');

% Input the initial value.
s(1)=0.;
s(2)=input('Input s(2) (second initial value): ');

% Display value of s(1) and s(2)
disp(s(1));
disp(s(2));

% Loop on all points
for n=2:1:m-1
  s(n+1)     = s(n) + s(n-1)^2;
  disp(s(n+1));
end

% Plots s(n) versus n
gset auto
plot(s,'+-g;s(n);');

disp('');
disp(s(m))
```

```
% Name:     esempio8.m
% Author:   Simone Zuccher
% Created:  12 Apr 2007
% Purpose:  Compute x(k+1) = log( 1 + x(k) )
% Input:    Number of total iterations and x(1)
% Output:   Plot of x(k) versus k and x(k+1) versus x(k)
% Modified:

% Clear all variables
clear

% Change format to long exponential
format long e

% Input the number of total iterations
m=input('Input N (number of iterations): ');

% Input the initial value.
s(1)=input('Input s(1) (initial value): ');

% Set s(1) positive if negative
if(s(1)<0)
  s(1)=abs(s(1));
endif

% Display value of s(1)
disp(s(1));

% Assign x and y needed for 2nd plot
x(1)=s(1);
y(1)=0.0;

% Loop on all points
for n=1:1:m-1
  s(n+1)     = log(1.+s(n));
  disp(s(n+1));
  x(2*n)     = s(n);
  y(2*n)     = s(n+1);
  x(2*n+1)   = s(n+1);
  y(2*n+1)   = s(n+1);
end

% Plots s(n) versus n
gset auto
plot(s,'+-g;s(n);');

% Wait for keypressed
disp('Please press a key to continue...');
pause();

disp(s(m))

% Plot f(x), x and path
t=linspace(-.5,2,500);
plot(t,log(1.+t),'-g;log(1+t);',t,t,'-b;x;',x,y,'+-');
```

```
% Name:     logisticplot.m
% Author:   Simone Zuccher
% Created:  17 Apr 2007
% Purpose:  Plot last p_max iterations of the logistic map
%           x(k+1) = A*x(k)*(1-x(k))
%           computed up to t_max  for A_min < A < A_max with x_0 = .1
% Input:    A, number of total iterations and x(1)
% Output:   Plot of bifuration diagram
% Modified:

% Clear all variables
clear

% Change format to long exponential
format long e

% Set A_min and A_max and number of A-values
A_min = 2.8;
A_max = 4.0;
n = 1000;

% Set maximum number of iterations (we need to stop sooner or later...)
t_max = 1000;

% Set how many iterations from the last are shown for each value of A
p_max = 100;

% Set the initial point (arbitrary)
x0 = 0.1;

% Create vector of A and matrix for final plot
A = linspace(A_min, A_max, n);
pop = zeros(p_max, n);

% Compute the population for each value of A
for k = 1:n
  x = population(A(k), x0, t_max);
  % Retain only the last p_max iterations
  pop(:, k) = x(t_max-p_max+1:t_max);
end

% Set no key
gset nokey;
% Set the title on x
gset xlabel 'A';
% Set the range of x
gset xrange[2.8:4]
% Generate the plot
plot(A, pop, 'b.');
```

```
% Name:     population.m
% Author:   Simone Zuccher
% Created:  17 Apr 2007
% Purpose:  Compute the logistic map
%           x(k+1) = A*x(k)*(1-x(k))
%           computed up to n  given A and  x_0
% Input:    A, , x_0 and number of total iterations n
% Output:   Array of x values
% Modified:

function x =  population(A, x0, n)
  x = zeros(n, 1);
  x(1) = x0;
  for k = 1:n-1
    x(k + 1) = A * x(k) * (1 - x(k));
  end
```

```
% Name:     logistic.m
% Author:   Simone Zuccher
% Created:  16 Apr 2007
% Purpose:  Compute x(k+1) = A*x(k)*(1-x(k))
% Input:    A, number of total iterations and x(1)
% Output:   Plot of x(k) versus k and x(k+1) versus x(k)
% Modified:

% Clear all variables
clear

% Change format to long exponential
format long e

% Input the paramter A
A=input('Input A>0 (parameter of x(k+1)=A*x(k)*(1-x(k)) ): ');
if(A<0)
  A=abs(A);
endif

% Input the number of total iterations
m=input('Input N (number of iterations): ');

% Input the initial value. If negative, it will be a random number
s(1)=input('Input 0<=s(1)<=1 (if <0 or >1 then random): ');

% Set s(1) random if out of range
if((s(1)>1) || (s(1)<0))
  s(1)=rand(1);
endif

% Display value of s(1)
disp(s(1));

% Assign x and y needed for 2nd plot
x(1)=s(1);
y(1)=0.0;

% Loop on all points
for n=1:1:m-1
  s(n+1)     = A*s(n)*(1 - s(n));
  disp(s(n+1));
  x(2*n)     = s(n);
  y(2*n)     = s(n+1);
  x(2*n+1)   = s(n+1);
  y(2*n+1)   = s(n+1);
end

% Plots s(n) versus n
gset auto
gset key left Left reverse
plot(s,'+-g;s(n);');

% Wait for keypressed
disp('Please press a key to continue...');
pause();

% Plot f(x), x and path
t=linspace(0,1,500);
plot(t,A*t.*(1-t),'-g;A*x*(1-x);',t,t,'-b;x;',x,y,'+-');

disp('Last 8 points:');
disp(s(m-7:m)');
```

```
% Name:     sys1.m
% Author:   Simone Zuccher
% Created:  03 May 2007
% Purpose:  Solve the ODE system
%           x'=y+x(x^2+y^2-1)
%           y'=-x+y(x^2+y^2-1)
%           given the initial condition x0 and y0
% Input:    Initial condition [x0 y0], final time tfinal
% Output:   1. plot of x(t) versus y(t) together with the vector field
%           2. plot time histories of x(t) and y(t)
% Modified:
%
% The equilibrium points are the following:
%
% [x = 0, y = 0],
%

% Clear all variables
clear all;

% Window ranges
xmin=-3;
xmax=3;
ymin=-3;
ymax=3;

% Equilibrium points
eq = [0 0];
disp('Equilibrium points:');
disp(eq);

% Set initial conditions
x0=input('Insert initial conditions [x0 y0]: ');

% Set final time for integration
tmax=input('Insert final time: ');


disp('Initial condition:');
disp(x0);

% Time parameters
tmin=0;
dt=.01;
% Create time
t = tmin:dt:tmax;

% dx and dy used only for vectors
dx=abs(xmax-xmin)/30;
dy=abs(ymax-ymin)/30;
% rescales vector size
scale=0.027*max(abs(xmax-xmin),abs(ymax-ymin));

% Definition of the dynamical system
function xdot=limitc(x, t)
  u = x(1);
  v = x(2);
  xdot(1) = v+u*(u^2+v^2-1);
  xdot(2) = -u+v*(u^2+v^2-1);
endfunction

__gnuplot_set__ nokey

setax=[xmin xmax ymin ymax];
axis(setax)

[X, Y] = meshgrid(xmin:dx:xmax, ymin:dy:ymax);
DX = Y+X.*(X.^2 + Y.^2 - 1);
```

```
DY = -X+Y.*(X.^2 + Y.^2 - 1);
L = sqrt(DX.^2 + DY.^2);
mytitle=["Phase portrait. Initial conditions: x0=" num2str(x0(1)) ", y0=\
         " num2str(x0(2))];
__gnuplot_set__ nokey
__gnuplot_set__ xlabel 'x(t)'
__gnuplot_set__ ylabel 'y(t)'
title(mytitle)

% Plot vector field
quiver(X, Y, scale*DX./L, scale*DY./L)
hold on;

% Plot all equilibrium points
plot( eq(:,1), eq(:,2), '*k')

x =  lsode("limitc", x0, t)';
plot( x(1,1), x(2,1), '*k', x(1,:), x(2,:), '-r')
hold off;

% Wait for keypressed
disp('Please press a key to continue...');
pause();
mytitle=["Time histories. Initial conditions: x0=" num2str(x0(1)) ", y0=\
         " num2str(x0(2))];
__gnuplot_set__ auto
__gnuplot_set__ xlabel 't'
__gnuplot_set__ ylabel 'x(t), y(t)'
title(mytitle)
__gnuplot_set__ key

% Plot time histories
plot( t, x(1,:), '-r;x(t);', t, x(2,:), '-g;y(t);')
```

```
% Name:     sys2.m
% Author:   Simone Zuccher
% Created:  03 May 2007
% Purpose:  Solve the ODE system
%           x'=y+x(cos(x^2+y^2)-1)(sin(x^2+y^2)-1)
%           y'=-x+y(cos(x^2+y^2)-1)(sin(x^2+y^2)-1)
%           given the initial condition x0 and y0
% Input:    Initial condition [x0 y0], final time tfinal
% Output:   1. plot of x(t) versus y(t) together with the vector field
%           2. plot time histories of x(t) and y(t)
% Modified:
%
% The equilibrium points are the following:
%
% [x = 0, y = 0],
%

% Clear all variables
clear all;

% Window ranges
xmin=-3;
xmax=3;
ymin=-3;
ymax=3;

% Equilibrium points
eq = [0 0];
disp('Equilibrium points:');
disp(eq);

% Set initial conditions
x0=input('Insert initial conditions [x0 y0]: ');

% Set final time for integration
tmax=input('Insert final time: ');


disp('Initial condition:');
disp(x0);

% Time parameters
tmin=0;
dt=.01;
% Create time
t = tmin:dt:tmax;

% dx and dy used only for vectors
dx=abs(xmax-xmin)/30;
dy=abs(ymax-ymin)/30;
% rescales vector size
scale=0.027*max(abs(xmax-xmin),abs(ymax-ymin));

% Definition of the dynamical system
function xdot=dsys(x, t)
  u = x(1);
  v = x(2);
  xdot(1) = v+u*(cos(u^2+v^2)-1)*(sin(u^2+v^2)-1);
  xdot(2) = -u+v*(cos(u^2+v^2)-1)*(sin(u^2+v^2)-1);
endfunction

__gnuplot_set__ nokey
axis([xmin xmax ymin ymax]);

[X, Y] = meshgrid(xmin:dx:xmax, ymin:dy:ymax);

DX = Y+X.*(cos(X.^2 + Y.^2) - 1).*(sin(X.^2 + Y.^2) - 1);
DY = -X+Y.*(cos(X.^2 + Y.^2) - 1).*(sin(X.^2 + Y.^2) - 1);
```

```
L = sqrt(DX.^2 + DY.^2);
mytitle=["Phase portrait. Initial conditions: x0=" num2str(x0(1)) \
         ", y0=" num2str(x0(2))];
__gnuplot_set__ nokey
__gnuplot_set__ xlabel 'x(t)'
__gnuplot_set__ ylabel 'y(t)'
title(mytitle)

% Plot vector field
quiver(X, Y, scale*DX./L, scale*DY./L)
hold on;

% Plot all equilibrium points
plot( eq(:,1), eq(:,2), '*k')

x =  lsode("dsys", x0, t)';
plot( x(1,1), x(2,1), '*k', x(1,:), x(2,:), '-r')
hold off;

% Wait for keypressed
disp('Please press a key to continue...');
pause();
mytitle=["Time histories. Initial conditions: x0=" num2str(x0(1)) ", y0=\
         " num2str(x0(2))];
__gnuplot_set__ auto
__gnuplot_set__ xlabel 't'
__gnuplot_set__ ylabel 'x(t), y(t)'
title(mytitle)
__gnuplot_set__ key

% Plot time histories
plot( t, x(1,:), '-r;x(t);', t, x(2,:), '-g;y(t);')
```

```
% Name:     sys3.m
% Author:   Simone Zuccher
% Created:  03 May 2007
% Purpose:  Solve the ODE system
%           x'=-y+x(x^2+y^2)cos(\pi/(x^2+y^2))
%           y'=x+y(x^2+y^2)cos(\pi/(x^2+y^2))
%           given the initial condition x0 and y0
% Input:    Initial condition [x0 y0], final time tfinal
% Output:   1. plot of x(t) versus y(t) together with the vector field
%           2. plot time histories of x(t) and y(t)
% Modified:
%
% The equilibrium points are the following:
%
% [x = 0, y = 0],
%

% Clear all variables
clear all;

% Window ranges
xmin=-3;
xmax=3;
ymin=-3;
ymax=3;

% Equilibrium points
eq = [0 0];
disp('Equilibrium points:');
disp(eq);

% Set initial conditions
x0=input('Insert initial conditions [x0 y0]: ');

% Set final time for integration
tmax=input('Insert final time: ');

disp('Initial condition:');
disp(x0);

% Time parameters
tmin=0;
dt=.01;
% Create time
t = tmin:dt:tmax;

% dx and dy used only for vectors
dx=abs(xmax-xmin)/30;
dy=abs(ymax-ymin)/30;
% rescales vector size
scale=0.027*max(abs(xmax-xmin),abs(ymax-ymin));

% Definition of the dynamical system
function xdot=dsys(x, t)
  u = x(1);
  v = x(2);
  xdot(1) = -v+u*(u^2+v^2)*cos(pi/(u^2+v^2));
  xdot(2) = u+v*(u^2+v^2)*cos(pi/(u^2+v^2));
endfunction

__gnuplot_set__ nokey
axis([xmin xmax ymin ymax]);

[X, Y] = meshgrid(xmin:dx:xmax, ymin:dy:ymax);

DX = -Y+X.*(X.^2 + Y.^2).*cos(pi./(X.^2 + Y.^2));
DY = X+Y.*(X.^2 + Y.^2).*cos(pi./(X.^2 + Y.^2));
L = sqrt(DX.^2 + DY.^2);
```

```
mytitle=["Phase portrait. Initial conditions: x0=" num2str(x0(1)) \
          ", y0=" num2str(x0(2))];
__gnuplot_set__ nokey
__gnuplot_set__ xlabel 'x(t)'
__gnuplot_set__ ylabel 'y(t)'
title(mytitle)

% Plot vector field
quiver(X, Y, scale*DX./L, scale*DY./L)
hold on;

% Plot all equilibrium points
plot( eq(:,1), eq(:,2), '*k')

x =  lsode("dsys", x0, t)';
plot( x(1,1), x(2,1), '*k', x(1,:), x(2,:), '-r')
hold off;

% Wait for keypressed
disp('Please press a key to continue...');
pause();
mytitle=["Time histories. Initial conditions: x0=" num2str(x0(1)) \
         ", y0=" num2str(x0(2))];
__gnuplot_set__ auto
__gnuplot_set__ xlabel 't'
__gnuplot_set__ ylabel 'x(t), y(t)'
title(mytitle)
__gnuplot_set__ key

% Plot time histories
plot( t, x(1,:), '-r;x(t);', t, x(2,:), '-g;y(t);')
```

```
% Name:     sys4.m
% Author:   Simone Zuccher
% Created:  03 May 2007
% Purpose:  Solve the ODE system (Van der Pool equation)
%           x'=-y-a*x*(y^2-1)
%           y'=x
%           given the initial condition x0 and y0
% Input:    Initial condition [x0 y0], final time tfinal
% Output:   1. plot of x(t) versus y(t) together with the vector field
%           2. plot time histories of x(t) and y(t)
% Modified:
%
% The equilibrium points are the following:
%
% [x = 0, y = 0],
%

% Clear all variables
clear all;
global a;

% Window ranges
xmin=-3;
xmax=3;
ymin=-3;
ymax=3;

% Equilibrium points
eq = [0 0];
disp('Equilibrium points:');
disp(eq);

% Set parameter a
a=input('Insert a: ');

% Set initial conditions
x0=input('Insert initial conditions [x0 y0]: ');

% Set final time for integration
tmax=input('Insert final time: ');

disp('Initial condition:');
disp(x0);

% Time parameters
tmin=0;
dt=.01;
% Create time
t = tmin:dt:tmax;

% dx and dy used only for vectors
dx=abs(xmax-xmin)/30;
dy=abs(ymax-ymin)/30;
% rescales vector size
scale=0.027*max(abs(xmax-xmin),abs(ymax-ymin));

% Definition of the dynamical system
function xdot=dsys(x, t)
  global a;
  u = x(1);
  v = x(2);
  xdot(1) = -v-a*u*(v^2-1);
  xdot(2) = u;
endfunction

__gnuplot_set__ nokey
axis([xmin xmax ymin ymax]);
```

```
[X, Y] = meshgrid(xmin:dx:xmax, ymin:dy:ymax);

DX = -Y-a*X.*(Y.^2-1);
DY = X;
L = sqrt(DX.^2 + DY.^2);
mytitle=["Phase portrait. Initial conditions: x0=" num2str(x0(1))\
         ", y0=" num2str(x0(2)) ". a=" num2str(a) "."];
__gnuplot_set__ nokey
__gnuplot_set__ xlabel 'x(t)'
__gnuplot_set__ ylabel 'y(t)'
title(mytitle)

% Plot vector field
quiver(X, Y, scale*DX./L, scale*DY./L)
hold on;

% Plot all equilibrium points
plot( eq(:,1), eq(:,2), '*k')

x =  lsode("dsys", x0, t)';
plot( x(1,1), x(2,1), '*k', x(1,:), x(2,:), '-r')
hold off;

% Wait for keypressed
disp('Please press a key to continue...');
pause();
mytitle=["Time histories. Initial conditions: x0=" num2str(x0(1))\
         ", y0=" num2str(x0(2)) ". a=" num2str(a) "."];
__gnuplot_set__ auto
__gnuplot_set__ xlabel 't'
__gnuplot_set__ ylabel 'x(t), y(t)'
title(mytitle)
__gnuplot_set__ key

% Plot time histories
plot( t, x(1,:), '-r;x(t);', t, x(2,:), '-g;y(t);')
```

```
% Name:     sys5.m
% Author:   Simone Zuccher
% Created:  03 May 2007
% Purpose:  Solve the ODE system
%           x'=y
%           y'=(1 - x^2 - y^2)y - x
%           given the initial condition x0 and y0
% Input:    Initial condition [x0 y0], final time tfinal
% Output:   1. plot of x(t) versus y(t) together with the vector field
%           2. plot time histories of x(t) and y(t)
% Modified:
%
% The equilibrium points are the following:
%
% [x = 0, y = 0],
%

% Clear all variables
clear all;

% Window ranges
xmin=-2;
xmax=2;
ymin=-2;
ymax=2;

% Equilibrium points
eq = [0 0];
disp('Equilibrium points:');
disp(eq);

% Set initial conditions
x0=input('Insert initial conditions [x0 y0]: ');

% Set final time for integration
tmax=input('Insert final time: ');


disp('Initial condition:');
disp(x0);

% Time parameters
tmin=0;
dt=.01;
% Create time
t = tmin:dt:tmax;

% dx and dy used only for vectors
dx=abs(xmax-xmin)/30;
dy=abs(ymax-ymin)/30;
% rescales vector size
scale=0.027*max(abs(xmax-xmin),abs(ymax-ymin));

% Definition of the dynamical system
function xdot=dsys(x, t)
  u = x(1);
  v = x(2);
  xdot(1) = v;
  xdot(2) = (1-u^2-v^2)*v - u;
endfunction

__gnuplot_set__ nokey
axis([xmin xmax ymin ymax]);

[X, Y] = meshgrid(xmin:dx:xmax, ymin:dy:ymax);

DX = Y;
DY = (1-X.^2 - Y.^2).*Y - X;
```

```
L = sqrt(DX.^2 + DY.^2);
mytitle=["Phase portrait. Initial conditions: x0=" num2str(x0(1))\
         ", y0=" num2str(x0(2)) "."];
__gnuplot_set__ nokey
__gnuplot_set__ xlabel 'x(t)'
__gnuplot_set__ ylabel 'y(t)'
title(mytitle)

% Plot vector field
quiver(X, Y, scale*DX./L, scale*DY./L)
hold on;

% Plot all equilibrium points
plot( eq(:,1), eq(:,2), '*k')

x =  lsode("dsys", x0, t)';
plot( x(1,1), x(2,1), '*k', x(1,:), x(2,:), '-r')
hold off;

% Wait for keypressed
disp('Please press a key to continue...');
pause();
mytitle=["Time histories. Initial conditions: x0=" num2str(x0(1))\
         ", y0=" num2str(x0(2)) "."];
__gnuplot_set__ auto
__gnuplot_set__ xlabel 't'
__gnuplot_set__ ylabel 'x(t), y(t)'
title(mytitle)
__gnuplot_set__ key

% Plot time histories
plot( t, x(1,:), '-r;x(t);', t, x(2,:), '-g;y(t);')
```

```
% Name:     myLV.m
% Author:   Simone Zuccher
% Created:  17 May 2007
% Purpose:  solve the Lotka-Volterra system
%           u'= au - buv
%           v'= - cv + duv
%           given u0 and v0
% Input:    see file
% Output:   1. plot of u(t) versus v(t) together with the vector field
%           2. plot time histories of u(t), v(t)
% Modified:
%
% The equilibrium points are the following, but we consider only u0 and v0
% non-negative
%
% [u = 0, v = 0],
%
%       c        a
% [u = ---, v = ---],
%       d        b
%

% Clear all variables
clear all;

% Window ranges
xmin=0;
xmax=300;
ymin=0;
ymax=300;

% Model constant
global aa;

% Give instructions
disp('');
disp('~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~');
disp('This script solves the classic Lotka-Volterra system:');
disp('  u''= au - buv');
disp('  v''= - cv + duv');
disp('given a, b, c, d all positive');
% Set initial conditions
aa=input('Insert constants [a b c d]: ');

% Equilibrium points
eq = [0 0];
eq = [eq; aa(3)/aa(4) aa(1)/aa(2)];
disp('Equilibrium points:');
disp(eq);


% Set initial conditions
x0=input('Insert initial conditions [x0 y0]: ');


% Set final time for integration
tmax=input('Insert final time: ');

disp('Initial condition:');
disp(x0);

% Time parameters
tmin=0;
dt=.01;
% Create time
t = tmin:dt:tmax;

% dx and dy used only for vectors
```

```
dx=abs(xmax-xmin)/30;
dy=abs(ymax-ymin)/30;
% rescales vector size
scale=0.027*max(abs(xmax-xmin),abs(ymax-ymin));

% Definition of the dynamical system
function xdot=dsys(x, t)
  global aa;
  u = x(1);
  v = x(2);
  xdot(1) = aa(1)*u - aa(2)*u*v;
  xdot(2) = -aa(3)*v + aa(4)*u*v;
endfunction

__gnuplot_set__ nokey

setax=[xmin xmax ymin ymax];
axis(setax)

[X, Y] = meshgrid(xmin:dx:xmax, ymin:dy:ymax);
DX = aa(1)*X - aa(2)*X.*Y;
DY = -aa(3)*Y + aa(4)*X.*Y;
L = sqrt(DX.^2 + DY.^2);
mytitle=["Phase portrait. Initial conditions: x0=" num2str(x0(1)) \
         ", y0=" num2str(x0(2))];
__gnuplot_set__ nokey
__gnuplot_set__ xlabel 'x(t)'
__gnuplot_set__ ylabel 'y(t)'
title(mytitle)

% Plot vector field
quiver(X, Y, scale*DX./L, scale*DY./L)
hold on;

% Plot all equilibrium points
plot( eq(:,1), eq(:,2), '*k')

x =  lsode("dsys", x0, t)';
plot( x(1,1), x(2,1), '*k', x(1,:), x(2,:), '-r')
hold off;

% Wait for keypressed
disp('Please press a key to continue...');
pause();
mytitle=["Time histories. Initial conditions: x0=" num2str(x0(1)) \
         ", y0=" num2str(x0(2))];
__gnuplot_set__ auto
__gnuplot_set__ xlabel 't'
__gnuplot_set__ ylabel 'x(t), y(t)'
title(mytitle)
__gnuplot_set__ key

% Plot time histories
plot( t, x(1,:), '-r;x(t);', t, x(2,:), '-g;y(t);')
```

```
% Name:     compcoop.m
% Author:   Simone Zuccher
% Created:  17 May 2007
% Purpose:  solve the general system
%           u'= au + buu + cuv
%           v'= dv + evv + fuv
%           given u0 and v0
% Input:    see file
% Output:   1. plot of u(t) versus v(t) together with the vector field
%           2. plot time histories of u(t) and  v(t)
% Modified:
%
% The equilibrium points are the following, but we consider only u0 and v0
% non-negative
%
%
% [u = 0, v = 0],
%
%       a
% [u = - -, v = 0],
%       b
%               d
% [u = 0, v = - -],
%               e
%      a e - c d        a f - b d
% [u = ---------, v = - ---------]
%      c f - b e        c f - b e



% Clear all variables
clear all;

% Window ranges
xmin=-.0;
xmax=120.;
ymin=-.0;
ymax=120.;

% Model constant
global aa;

% Give instructions
disp('');
disp('~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~');
disp('This script solves the general system:');
disp('  u''= au + buu + cuv');
disp('  v''= dv + evv + fuv');
disp('given a, b, c, d, e, f.');
% Set initial conditions
%aa=input('Insert constants [a b c d e f]: ');
aa=[10 -0.1 -0.097 10 -(0.1-0.002) -0.097]


% Equilibrium points
eq = [0 0];
if(abs(aa(2))>0)
   eq = [eq; -aa(1)/aa(2) 0];
endif
if(abs(aa(5))>0)
   eq = [eq; 0 -aa(4)/aa(5)];
endif
if(abs(aa(3)*aa(6)-aa(2)*aa(5))>0)
   eq = [eq; (aa(1)*aa(5)-aa(3)*aa(4))/(aa(3)*aa(6)-aa(2)*aa(5)) \
             -(aa(1)*aa(6)-aa(2)*aa(4))/(aa(3)*aa(6)-aa(2)*aa(5)) ];
endif
disp('Equilibrium points:');
disp(eq);
```

```
% Set initial conditions
%x0=input('Insert initial conditions [x0 y0]: ');
x0=[.1 115];

% Set final time for integration
%tmax=input('Insert final time: ');
tmax=80;

disp('Initial condition:');
disp(x0);

% Time parameters
tmin=0;
dt=.01;
% Create time
t = tmin:dt:tmax;

% dx and dy used only for vectors
dx=abs(xmax-xmin)/30;
dy=abs(ymax-ymin)/30;
% rescales vector size
scale=0.027*max(abs(xmax-xmin),abs(ymax-ymin));

% Definition of the dynamical system
function xdot=dsys(x, t)
  global aa;
  u = x(1);
  v = x(2);
  xdot(1) = aa(1)*u + aa(2)*u*u + aa(3)*u*v;
  xdot(2) = aa(4)*v + aa(5)*v*v + aa(6)*u*v;
endfunction

__gnuplot_set__ nokey

setax=[xmin xmax ymin ymax];
axis(setax)

[X, Y] = meshgrid(xmin:dx:xmax, ymin:dy:ymax);
DX = aa(1)*X + aa(2)*X.*X + aa(3)*X.*Y;
DY = aa(4)*Y + aa(5)*Y.*Y + aa(6)*X.*Y;
L = sqrt(DX.^2 + DY.^2);
mytitle=["Phase portrait. Initial conditions: x0=" num2str(x0(1)) \
         ", y0=" num2str(x0(2))];
__gnuplot_set__ nokey
__gnuplot_set__ xlabel 'x(t)'
__gnuplot_set__ ylabel 'y(t)'
title(mytitle)

% Plot vector field
quiver(X, Y, scale*DX./L, scale*DY./L)
hold on;

% Plot all equilibrium points
plot( eq(:,1), eq(:,2), '*k')

x =  lsode("dsys", x0, t)';
plot( x(1,1), x(2,1), '*k', x(1,:), x(2,:), '-r')
hold off;
oct2ps("excom5pp");

% Wait for keypressed
disp('Please press a key to continue...');
pause();
mytitle=["Time histories. Initial conditions: x0=" num2str(x0(1)) \
         ", y0=" num2str(x0(2))];
__gnuplot_set__ auto
```

```
__gnuplot_set__ xlabel 't'
__gnuplot_set__ ylabel 'x(t), y(t)'
title(mytitle)
__gnuplot_set__ key

% Plot time histories
plot( t, x(1,:), '-r;x(t);', t, x(2,:), '-g;y(t);')
oct2ps("excom5th");
```

```
% Name:     LVmod.m
% Author:   Simone Zuccher
% Created:  17 May 2007
% Purpose:  solve the modified Lotka-Volterra system
%           u'=u(1-u)-kuv^2
%           v'=v(1-v)-kvu^2
%           given u0 and v0, and k>3/4
% Input:    see file
% Output:   1. plot of u(t) versus v(t) together with the vector field
%           2. plot time histories of u(t), v(t)
% Modified:
%
% The equilibrium points are the following, but we consider only u0 and v0
% non-negative
%
% [u = 0, v = 0],
%
% [u = 1, v = 0],
%
% [u = 0, v = 1],
%
%        sqrt(4 k + 1) + 1        sqrt(4 k + 1) + 1
% [u = - ----------------, v = - ----------------],
%              2 k                      2 k
%
%      sqrt(4 k + 1) - 1      sqrt(4 k + 1) - 1
% [u = ----------------, v = ----------------],
%             2 k                    2 k
%
%      sqrt(4 k - 3) + 1        sqrt(4 k - 3) - 1
% [u = ----------------, v = - ----------------],
%             2 k                      2 k
%
%        sqrt(4 k - 3) - 1      sqrt(4 k - 3) + 1
% [u = - ----------------, v = ----------------]
%               2 k                    2 k
%

% Clear all variables
clear all;

% Window ranges
xmin=0;
xmax=1;
ymin=0;
ymax=1;


% Model constant
global k;
% Set model constant
k=input('Insert k: ');

% Equilibrium points
eq = [0 0];
eq = [eq; 0 1];
eq = [eq; 1 0];
eq= [eq; -(sqrt(4 * k + 1) + 1)/2/k -(sqrt(4 * k + 1) + 1)/2/k];
eq= [eq; (sqrt(4 * k + 1) - 1)/2/k (sqrt(4 * k + 1) - 1)/2/k];
eq= [eq; (sqrt(4 * k - 3) + 1)/2/k -(sqrt(4 * k - 3) - 1)/2/k];
eq= [eq; -(sqrt(4 * k - 3) - 1)/2/k (sqrt(4 * k - 3) + 1)/2/k];
disp('Equilibrium points:');
disp(eq);


% Set initial conditions
x0=input('Insert initial conditions [x0 y0]: ');
```

```
% Set final time for integration
tmax=input('Insert final time: ');

disp('Initial condition:');
disp(x0);

% Time parameters
tmin=0;
dt=.01;
% Create time
t = tmin:dt:tmax;

% dx and dy used only for vectors
dx=abs(xmax-xmin)/30;
dy=abs(ymax-ymin)/30;
% rescales vector size
scale=0.027*max(abs(xmax-xmin),abs(ymax-ymin));



% Definition of the dynamical system
function xdot=dsys(x, t)
  global k;
  u = x(1);
  v = x(2);
  xdot(1) = u*(1-u)-k*u*v^2;
  xdot(2) = v*(1-v)-k*v*u^2;
endfunction

__gnuplot_set__ nokey

setax=[xmin xmax ymin ymax];
axis(setax)

[X, Y] = meshgrid(xmin:dx:xmax, ymin:dy:ymax);

DX = X.*(1-X)-k*X.*(Y.^2) ;
DY = Y.*(1-Y)-k*Y.*(X.^2);
L = sqrt(DX.^2 + DY.^2);
mytitle=["Phase portrait. Initial conditions: x0=" num2str(x0(1)) \
         ", y0=" num2str(x0(2))];
__gnuplot_set__ nokey
__gnuplot_set__ xlabel 'x(t)'
__gnuplot_set__ ylabel 'y(t)'
title(mytitle)

% Plot vector field
quiver(X, Y, scale*DX./L, scale*DY./L)
hold on;

% Plot all equilibrium points
plot( eq(:,1), eq(:,2), '*k')

x =  lsode("dsys", x0, t)';
plot( x(1,1), x(2,1), '*k', x(1,:), x(2,:), '-r')
hold off;

% Wait for keypressed
disp('Please press a key to continue...');
pause();
mytitle=["Time histories. Initial conditions: x0=" num2str(x0(1)) \
         ", y0=" num2str(x0(2))];
__gnuplot_set__ auto
__gnuplot_set__ xlabel 't'
__gnuplot_set__ ylabel 'x(t), y(t)'
title(mytitle)
__gnuplot_set__ key
```

```
% Plot time histories
plot( t, x(1,:), '-r;x(t);', t, x(2,:), '-g;y(t);')
```

```
# Name:     surface
# Author:   Simone Zuccher
# Created:  24 May 2007
# Purpose:  plot the stability bound for the realistic Lotka-Volterra system
#           u'= u(1-u) - auv/(u+d)
#           v'= bv(1 - v/u)
# Modified:
#
reset
set xlabel '$a$'
set ylabel '$c$'
set zlabel '$b$' -7,-7
set view 64, 207
set pm3d
set palette gray
set isosamples 50, 50
unset surface
set contour s
set cntrparam levels discrete 1e-6, .1, .2, .4, .5, .8
set ztics 0,.2
set cbtics 0,.2
set key 11, -1.3
surf(x,y) = ((x-sqrt((1-x-y)**2+4*y))*(1+x+y-sqrt((1-x-y)**2+4*y))/2./x >=0) \
            ? (x-sqrt((1-x-y)**2+4*y))*(1+x+y-sqrt((1-x-y)**2+4*y))/2./x : 0
splot [0:5] [0:1] surf(x,y) t 'Stability surface' w l
```

```
% Name:     realLV.m
% Author:   Simone Zuccher
% Created:  24 May 2007
% Purpose:  solve the realistic Lotka-Volterra system
%           u'= u(1-u) - auv/(u+d)
%           v'= bv(1 - v/u)
%           given u0 and v0
% Input:    see file
% Output:   1. plot of u(t) versus v(t) together with the vector field
%           2. plot time histories of u(t), v(t)
% Modified:
%
% The non-negative equilibrium points are the following
%
%                             2                 2
%       SQRT(d  + (2 a + 2) d + a  - 2 a + 1) - d - a + 1
% [ u = ------------------------------------------------,
%                               2
%
%                             2                 2
%       SQRT(d  + (2 a + 2) d + a  - 2 a + 1) - d - a + 1
%   v = ------------------------------------------------ ]
%                               2
%

% Clear all variables
clear all;

% Window ranges
xmin=0;
xmax=1;
ymin=0;
ymax=.5;

% Model constant
global aa;

% Give instructions
disp('');
disp('~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~');
disp('This script solves a real version of the Lotka-Volterra system:');
disp('  u''= u(1-u) - auv/(u+d)');
disp('  v''= bv(1 - v/u)');
disp('given a, b, d ');

% Set initial conditions
aa=input('Insert constants [a b d]: ');

% Compute b_stab (b<b_stab ensures stability)
bstab=(aa(1)-sqrt((1-aa(1)-aa(3))^2+4*aa(3)))*\
      (1+aa(1)+aa(3)-sqrt((1-aa(1)-aa(3))^2+4*aa(3)))/(2*aa(1));

if (aa(2)<bstab)
   disp('WARNING: a, b, d out of the stability region');
endif

% Equilibrium points
eq = [ (sqrt(aa(3)^2+(2*aa(1)+2)*aa(3)+aa(1)^2-2*aa(1)+1)-aa(3)-aa(1)+1)/2\
       (sqrt(aa(3)^2+(2*aa(1)+2)*aa(3)+aa(1)^2-2*aa(1)+1)-aa(3)-aa(1)+1)/2];
%eq = [eq; aa(3)/aa(4) aa(1)/aa(2)];
disp('Equilibrium points:');
disp(eq);


% Set initial conditions
x0=input('Insert initial conditions [x0 y0]: ');

% Set final time for integration
```

```
tmax=input('Insert final time: ');

disp('Initial condition:');
disp(x0);

% Time parameters
tmin=0;
dt=.01;
% Create time
t = tmin:dt:tmax;

% dx and dy used only for vectors
dx=abs(xmax-xmin)/30;
dy=abs(ymax-ymin)/30;
% rescales vector size
scale=0.027*max(abs(xmax-xmin),abs(ymax-ymin));

% Definition of the dynamical system
function xdot=dsys(x, t)
  global aa;
  u = x(1);
  v = x(2);
  xdot(1) = u*(1-u) - aa(1)*u*v/(u+aa(3));
  xdot(2) = aa(2)*v*(1 - v/u);
endfunction

__gnuplot_set__ nokey

setax=[xmin xmax ymin ymax];
axis(setax)

[X, Y] = meshgrid(xmin:dx:xmax, ymin:dy:ymax);
DX = X.*(1-X) - aa(1).*X.*Y./(X+aa(3));
DY = aa(2).*Y.*(1 - Y./X);
L = sqrt(DX.^2 + DY.^2);
mytitle=["Phase portrait. Initial conditions: x0=" num2str(x0(1)) \
         ", y0=" num2str(x0(2)) \
 "; a=" num2str(aa(1)) ", b=" num2str(aa(2))\
 ", c=" num2str(aa(3))];
__gnuplot_set__ nokey
__gnuplot_set__ xlabel 'x(t)'
__gnuplot_set__ ylabel 'y(t)'
title(mytitle)

% Plot vector field
quiver(X, Y, scale*DX./L, scale*DY./L)
hold on;

% Plot all equilibrium points
plot( eq(:,1), eq(:,2), '*k')

x =  lsode("dsys", x0, t)';
plot( x(1,1), x(2,1), '*k', x(1,:), x(2,:), '-r')
hold off;

% Wait for keypressed
disp('Please press a key to continue...');
pause();
mytitle=["Time histories. Initial conditions: x0=" num2str(x0(1)) \
         ", y0=" num2str(x0(2)) \
 "; a=" num2str(aa(1)) ", b=" num2str(aa(2))\
 ", c=" num2str(aa(3))];
__gnuplot_set__ auto
__gnuplot_set__ xlabel 't'
__gnuplot_set__ ylabel 'x(t), y(t)'
title(mytitle)
__gnuplot_set__ key
```

```
% Plot time histories
plot( t, x(1,:), '-r;x(t);', t, x(2,:), '-g;y(t);')
```

```
% Name:     tp.m
% Author:   Simone Zuccher
% Created:  24 May 2007
% Purpose:  solve the threshold phenomena system
%           u'= u(1+ 1/(1+(u-2)^2) - v)
%           v'= v(u - (v+1))
%           given u0 and v0
% Input:    see file
% Output:   1. plot of u(t) versus v(t) together with the vector field
%           2. plot time histories of u(t), v(t)
% Modified:
%
% The non-negative equilibrium points are the following
%
% [u = 2.353209961160612, v = 1.353209961160612]
%

% Clear all variables
clear all;

% Window ranges
xmin=0;
xmax=5;
ymin=0;
ymax=3;


% Equilibrium points
eq = [ 2.682327816337427 1.682327816337427];
disp('Equilibrium points:');
disp(eq);


% Set initial conditions
x0=input('Insert initial conditions [x0 y0]: ');

% Set final time for integration
tmax=input('Insert final time: ');

disp('Initial condition:');
disp(x0);

% Time parameters
tmin=0;
dt=.01;
% Create time
t = tmin:dt:tmax;

% dx and dy used only for vectors
dx=abs(xmax-xmin)/30;
dy=abs(ymax-ymin)/30;
% rescales vector size
scale=0.027*max(abs(xmax-xmin),abs(ymax-ymin));

% Definition of the dynamical system
function xdot=dsys(x, t)
  u = x(1);
  v = x(2);
  xdot(1) = u*(1+ 1/(1+(u-2)^2) - v);
  xdot(2) = v*(u - (v+1));
endfunction

__gnuplot_set__ nokey

setax=[xmin xmax ymin ymax];
axis(setax)

[X, Y] = meshgrid(xmin:dx:xmax, ymin:dy:ymax);
```

```
DX = X.*(1+ 1./(1+(X-2).^2) - Y);
DY = Y.*(X - (Y+1));
L = sqrt(DX.^2 + DY.^2);
mytitle=["Phase portrait. Initial conditions: x0=" num2str(x0(1)) \
         ", y0=" num2str(x0(2))];
__gnuplot_set__ nokey
__gnuplot_set__ xlabel 'x(t)'
__gnuplot_set__ ylabel 'y(t)'
title(mytitle)

% Plot vector field
quiver(X, Y, scale*DX./L, scale*DY./L)
hold on;

% Plot all equilibrium points
plot( eq(:,1), eq(:,2), '*k')

x =  lsode("dsys", x0, t)';
plot( x(1,1), x(2,1), '*k', x(1,:), x(2,:), '-r')
hold off;

% Wait for keypressed
disp('Please press a key to continue...');
pause();
mytitle=["Time histories. Initial conditions: x0=" num2str(x0(1)) \
         ", y0=" num2str(x0(2))];
__gnuplot_set__ auto
__gnuplot_set__ xlabel 't'
__gnuplot_set__ ylabel 'x(t), y(t)'
title(mytitle)
__gnuplot_set__ key

% Plot time histories
plot( t, x(1,:), '-r;x(t);', t, x(2,:), '-g;y(t);')
```

```
% Name:     mylorenz.m
% Author:   Simone Zuccher
% Created:  24 May 2007
% Purpose:  solve the classic Lorenz system
%           u'= a(v-u)
%           v'= -uw+bu -v
%           z'= uv-cw
%           given u0 and v0
% Input:    see file
% Output:   1. plot of u(t), v(t), w(t)
%           2. plot time histories of u(t), v(t), w(t)
% Modified:
%
% The non-negative equilibrium points are the following
%
% [u = 0, v = 0, w = 0]
%
% [u = SQRT(b c - c), v = SQRT(b - 1) SQRT(c), w = b - 1],
%
% [u = - SQRT(b c - c), v = - SQRT(b - 1) SQRT(c), w = b - 1],
%
%

% Clear all variables
clear all;

% Model constant
global aa;

% Set initial conditions
aa=input('Insert coefficients [a b c]: ');
%aa=[10 28 8./3.]

% Equilibrium points
eq = [ 0 0 0];
eq = [eq; sqrt(aa(2)*aa(3)-aa(3)) sqrt(aa(2)-1)*sqrt(aa(3)) aa(2)-1];
eq = [eq; -sqrt(aa(2)*aa(3)-aa(3)) -sqrt(aa(2)-1)*sqrt(aa(3)) aa(2)-1];

disp('Equilibrium points:');
disp(eq);


% Set initial conditions
x0=input('Insert initial conditions [x0 y0]: ');
%x0=[3 15 1];

% Set final time for integration
tmax=input('Insert final time: ');
%tmax=100;

disp('Initial condition:');
disp(x0);

% Time parameters
tmin=0;
dt=.01;
% Create time
t = tmin:dt:tmax;

% Definition of the dynamical system
function xdot=dsys(x, t)
  global aa;
  u = x(1);
  v = x(2);
  w = x(3);
  xdot(1) = aa(1)*(v-u);
  xdot(2) = -u*w+aa(2)*u - v;
  xdot(3) = u*v-aa(3)*w;
```

```
endfunction

x =  lsode("dsys", x0, t');

mytitle=["Phase portrait. Initial conditions: x0=" num2str(x0(1)) \
         ", y0=" num2str(x0(2)) ", z0=" num2str(x0(3)) \
 ", a=" num2str(aa(1)) ", b=" num2str(aa(2)) \
 ", c=" num2str(aa(3))];
__gnuplot_set__ nokey
__gnuplot_set__ xlabel 'x(t)'
__gnuplot_set__ ylabel 'y(t)'
__gnuplot_set__ zlabel 'z(t)'
__gnuplot_set__ parametric
__gnuplot_set__ view
%__gnuplot_set__ view 120, 30
title(mytitle)
gsplot x
```