Equazioni Differenziali Ordinarie

Mario Ambrosino

11 Giugno 2018

Indice

Capitolo 1

Equazioni differenziali ordinarie

Seguirò [1, 2] per lo studio e l'implementazioni di algoritmi atti a risolvere equazioni differenziali ordinarie ai valori iniziali.

1.1 Problemi ai valori iniziali

Il comportamento di un sistema dinamico può essere descritto tramite un problema di Cauchy con condizioni iniziali assegnate. Il sistema dinamico può essere ricondotto ad un sistema di equazioni differenziali del primo ordine

$$\frac{d\boldsymbol{y}}{dt} = \boldsymbol{g}\left(\boldsymbol{y}, t\right) \tag{1.1}$$

dove $y = (y_1, \ldots y_l)$ è il vettore delle variabili dinamiche e

$$\boldsymbol{g}\left(\boldsymbol{y}, t\right) = \left(g_1\left(\boldsymbol{y}, t\right), \ldots, g_l\left(\boldsymbol{y}, t\right)\right) \tag{1.2}$$

è il vettore delle velocità generalizzate. Ad esempio, nel caso di una particella in moto all'interno di un campo di forze elastiche in uno spazio 1-dim, bisogna considerare l'espressione dell'equazione di Newton (del secondo ordine)

$$f = ma \tag{1.3}$$

e tradurlo nel sistema dinamico composto in questo caso da due equazioni:

$$\frac{dy_1}{dt} = y_2, \tag{1.4}$$

$$\frac{dy_2}{dt} = -\frac{k}{m} y_1; \tag{1.5}$$

Consideriamo problemi 1-dim, è semplice l'estensione al caso a più dimensioni.

1.1.1 Metodo di Eulero

Consideriamo di approssimare la derivata con il rapporto incrementale (equivalente alla formula a due punti):

$$\frac{dy}{dt} \approx \frac{y_{i+1} - y_i}{t_{i+1} - t_i} \approx g\left(y_i, t_i\right). \tag{1.6}$$

In tal caso si ha che l'errore ad ogni iterazione è confrontabile con il timestep: $\mathcal{O}\left(|t_{i+1} - t_i|\right)$. Definendo $\tau \equiv t_{i+1} - t_i$ si ottiene la formula ricorsiva

$$y_{i+1} = y_i + g_i \tau + \mathcal{O}\left(\tau^2\right) \tag{1.7}$$

note come *Metodo di Eulero.* Osserviamo che possiamo riscrivere il problema nell'equazione 1.1 in forma integrata:

$$y_{i+j} = y_i + \int_{t_i}^{t_{i+j}} g\left(y, t\right)\, dt. \tag{1.8}$$

L'approssimazione della funzione integranda determina il tipo di metodo utilizzato per la risoluzione della ODE considerata: in particolare per $j = 1$ e $g\left(y, t\right) \simeq g_i$ si riottiene l'algoritmo di integrazione di Eulero.

1.1.2 Metodo di Picard

Consideriamo lo schema adattivo in cui, nell'espressione integrata che riportiamo

$$y_{i+j} = y_i + \int_{t_i}^{t_{i+j}} g\left(y, t\right)\, dt. \tag{1.9}$$

si valuta numericamente l'integranda a destra con le soluzioni ottenute tramite un altro metodo (ad esempio, un altra iterazione del metodo di Picard oppure il metodo di Eulero stesso). Immaginiamo di avere $j = 1$. Allora ad esempio fissando $y_{i+1}^{(0)} = y_i$ e utilizzando la regola di

integrazione di numerica del trapezio si ha che:

$$y_{i+1}^{(0)} = y_i + \frac{\tau}{2}\left(g_i + g_{i+1}\right) + \mathcal{O}\left(\tau^3\right) \tag{1.10}$$

Il metodo di Picard può essere lento se la scelta della funzione iniziale è poco opportuna

1.1.3 Metodi "Predictor - Corrector"

L'idea di base dei metodi "Predictor-Corrector" è di utilizzare due algoritmi di risoluzione per ODE in sequenza: il primo, esplicito e meno accurato, viene utilizzato per predire il valore y_{i+1} della sequenza, mentre il secondo, implicito e più accurato, viene utilizzato per correggere il valore di y_{i+1}.

Osservazione 1. Tutti questi metodi possono essere migliorati implementando una quadratura migliore, ad esempio passando da regola del trapezio a Simpson.

1.1.4 Metodo di Runge-Kutta

Espandiamo $y\left(t+\tau\right)$ in serie di τ attorno t. Si ha allora, ricordando che $\frac{dy}{dt} = g\left(y,t\right)$, che

$$\begin{aligned} y\left(t+\tau\right) &= y + \tau y' + \frac{\tau^2}{2}y'' + \frac{\tau^3}{3!}y^{(3)} + \dots \\ &= y + \tau g + \frac{\tau^2}{2}\left(g_t + g g_y\right) + \frac{\tau^3}{6}\left(g_{tt} + g_t g_y + g g_y^2 + 2 g g_{ty} + g^2 g_{yy}\right) + \dots \end{aligned}$$

Possiamo notare che è possibile scrivere la soluzione di ordine $m - mo$ in modo formale come

$$y\left(t+\tau\right) = y\left(t\right) + \alpha_1 c_1 + \alpha_2 c_2 + \dots + \alpha_m c_m \tag{1.11}$$

dove

$$c_1 = \tau g\left(y,t\right), \tag{1.12}$$

$$c_2 = \tau g\left(y + \nu_{21} c_1,\, t + \nu_{21}\tau\right), \tag{1.13}$$

$$\vdots \tag{1.14}$$

$$c_m = \tau g\left(y + \sum_{i=1}^{m-1} \nu_{mi} c_i,\, t + \tau \sum_{i=1}^{m-1} \nu_{mi}\right), \tag{1.15}$$

dove i parametri $\{\alpha_i\}_{i=1,2,\ldots,m}$ e $\{\nu_{ij}\}_{\substack{i=1,2,\ldots,m \\ j<i}}$ sono parametri da determinare. Si può così confrontare un'espansione in serie di taylor dei coefficienti c_i in funzione di τ con tale espansione al fine di ottenere l'espressione dei parametri $\{\alpha_i, \nu_{ij}\}$.

Determinazione parametri Runge Kutta caso $m = 2$ Si ha in tal caso

$$y(t+\tau) = y(t) + \alpha_1 c_1 + \alpha_2 c_2 + \mathcal{O}(\tau^2) \tag{1.16}$$

$$c_1 = \tau g(y,t) \tag{1.17}$$

$$c_2 = \tau g(y + \nu_{21} c_1,\, t + \nu_{21}\tau) \tag{1.18}$$

e al contempo $y(t+\tau) = y + \tau g + \frac{\tau^2}{2}(g_t + g g_y)$. Allora

$$c_2(y,t) = \tau g + \nu_{21}\tau^2 (g_t + g g_y) \tag{1.19}$$

dunque

$$y(t+\tau) \approx y(t) + \alpha_1 c_1 + \alpha_2 c_2 = y(t) + \alpha_1 \tau g + \alpha_2 \left(\tau g + \nu_{21}\tau^2 (g_t + g g_y)\right) \tag{1.20}$$

dal confronto risultano dunque i seguenti vincoli.

$$\begin{cases} \alpha_1 + \alpha_2 = 1 \\ \alpha_2 \nu_{21} = 1/2 \end{cases} \tag{1.21}$$

possiamo allora scegliere $\alpha_1 = \alpha_2 = 1/2$ e $\nu_{21} = 1$

Parametri Runge Kutta caso $m = 4$ Il metodo standard utilizzato per l'integrazione di equazioni numeriche è in realtà è l'algoritmo di Runge Kutta di ordine 4

$$y(t+\tau) = y(t) + \frac{1}{6}(c_1 + 2c_2 + 2c_3 + c_4) \tag{1.22}$$

$$c_1 = \tau g(y,t) \tag{1.23}$$

$$c_2 = \tau g\left(y + \frac{c_1}{2},\, t + \frac{\tau}{2}\right) \tag{1.24}$$

$$c_3 = \tau g\left(y + \frac{c_2}{2},\, t + \frac{\tau}{2}\right) \tag{1.25}$$

$$c_4 = \tau g(y + c_3,\, t + \tau) \tag{1.26}$$

Capitolo 2

Implementazioni degli algoritmi proposti

Ho realizzato un listato per la simulazione della dinamica di un sistema del secondo ordine partendo dall'oscillatore armonico semplice, passando quindi per il caso smorzato e forzato per poi osservare la dinamica del pendolo forzato alla ricerca euristica delle condizioni di caoticità del moto.

2.1 Oscillatore Armonico Semplice

Il listato sull'oscillatore armonico è stato utilizzato anche per osservare l'andamento dell'errore nei confronti della soluzione analitica. Dato un'oscillatore caratterizzato da frequenza ω_0, la sua legge dinamica è data da:

$$\frac{d^2 y\,(t)}{dt^2} = -k\, y\,(t) \tag{2.1}$$

Assegnate le velocità y_0 e v_o si ha che la soluzione analitica è data da

$$y\,(t) = \left(y_0^2 + \frac{v_0^2}{\omega_0^2}\right)^{1/2} \cos\left[\omega_0 t - \arctan\left(-\frac{v_0}{\omega_0 x_0}\right)\right] \tag{2.2}$$

Il listato per la soluzione numerica, `harmonic_oscillator.c`, è stato caricato in `https://github.com/mario-ambrosino/NODE/blob/master/sources/harmonic_oscillator.c`. Come anticipato, abbiamo utilizzato questo modello per valutare la stabilità rispetto agli errori dei vari metodi. In realtà come al solito consiglio di utilizzare lo script associato per un più rapido confronto tra i vari metodi, con i parametri già impostati nello script `solve_harmonic_oscillator.sh` nella cartella principale su GitHub. (cfr. **Figura 2.1**)

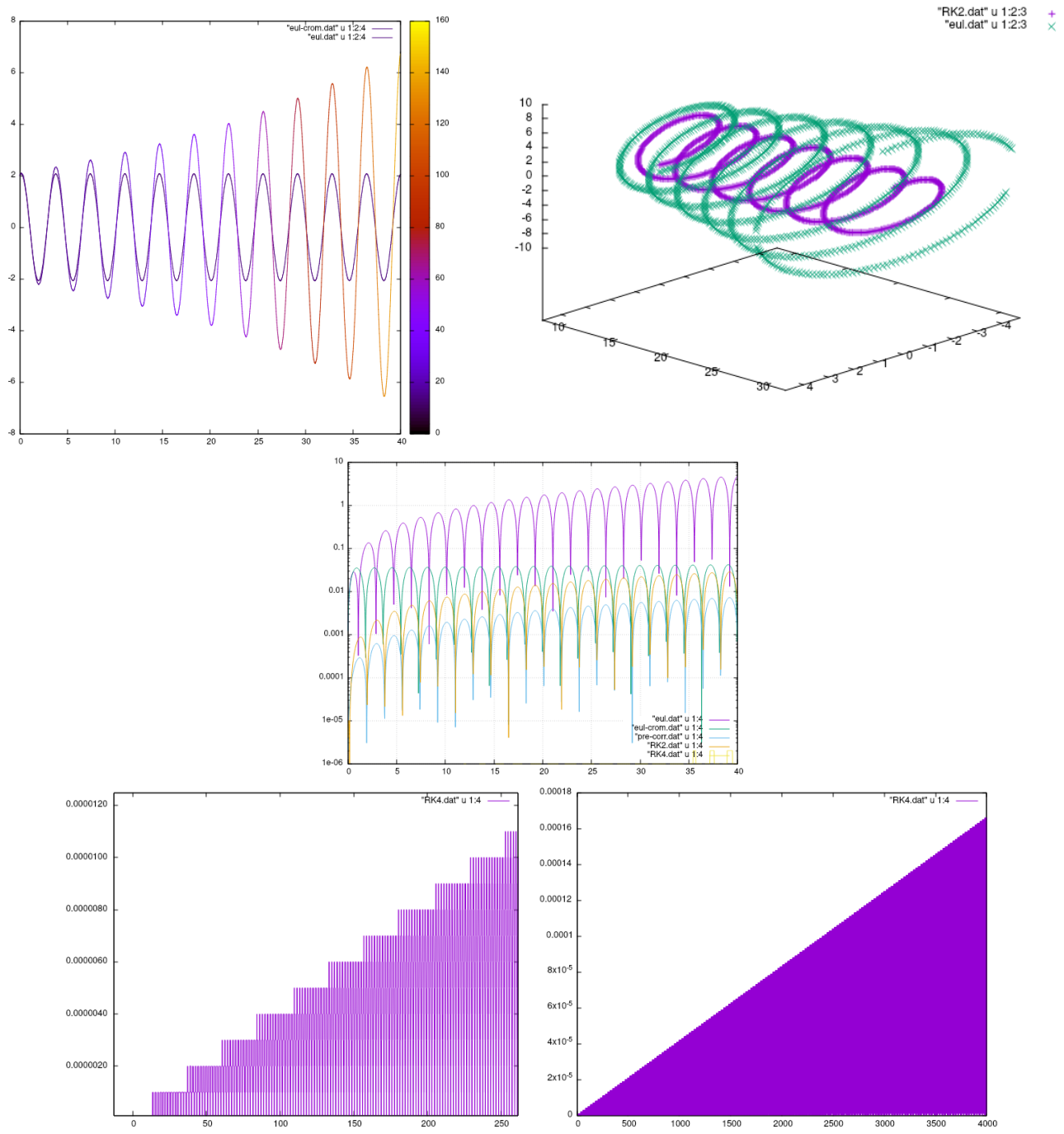


Figura 2.1: `harmonic_oscillator.c`, In alto: confronto tra evoluzione del tempo di Eulero ed Eulero-Cromer e confronto tra spazio delle fasi di Eulero e Runge-Kutta di ordine 2. Al centro: grafico semilogaritmico dell'andamento della discrepanza tra soluzione numerica e analitica. Da notare il piccolissimo valore del Runge - Kutta di ordine 4. In basso: andamento delle discrepanze per Runge-Kutta ordine 4 per diversi range: da notare l'andamento lineare, differente dall'andamento oscillante di tutti gli altri algoritmi.

2.2 Oscillatore Armonico Smorzato e Forzato

Ho dapprima implementato il solo oscillatore smorzato, per poi passare a quello forzato: la differenza sta nella definizione di una funzione dinamica (che restituisce il valore della forza per ogni punto dello spazio delle fasi ad un dato istante), che è scalata dall'essere soltanto dipendente dalla posizione, all'essere autonoma ma dipendente anche dalla velocità fino ad essere anche dipendente dal tempo. I due sorgenti di interesse sono ‘‘`damped_harmonic_oscillator`’’ e ‘‘`forced_harmonic_oscillator`’’. Essi sono associati rispettivamente alle equazioni differenziali:

$$\frac{d^2y\,(t)}{dt^2} = -\,k\,y\,(t) - \beta\,\frac{dy\,(t)}{dt} \tag{2.3}$$

$$\frac{d^2y\,(t)}{dt^2} = -\,k\,y\,(t) - \beta\,\frac{dy\,(t)}{dt} + A\cos\left(\omega_0 t\right) \tag{2.4}$$

Analizziamo direttamente (cfr. **Figura 2.2**) qualche caso riguardante il modello più generale, che contiene naturalmente il modello smorzato con $\omega_0 = 0$: 1) $k = 3000$, $\beta = 0$, $\omega_0 = 0$, $A = 0$; e 2) $k = 3000$, $\beta = 10$, $\omega_0 = 0$, $A = 0$; i valori successivi generano un intervallo equispaziato centrato attorno la condizione di risonanza dell'oscillatore, i cui parametri sono i seguenti: 3) $k = 3000$, $\beta = 10$, $\omega_0 = 54.31390$, $A = 1000$;

2.3 Pendolo Semplice e Smorzato: Pendolo Caotico

Il sorgente di interesse è ‘‘`pendulum.c`’’. Per la ricerca della regione caotica invece consiglio l'utlizzo dello script ‘‘`chaotic_pendulum.sh`’’. Essi sono associati rispettivamente alle equazioni differenziali:

$$\frac{d^2\theta\,(t)}{dt^2} = -\,k\,\sin(\theta\,(t)) - \beta\,\frac{d\theta\,(t)}{dt} + A\cos\left(\omega_0 t\right) \tag{2.5}$$

In questo caso riporto semplicemente il caso della ricerca della condizione di moto caotico. Utilizzo come parametri attorno cui variare $\omega_0 \in [0.9, 1.99]$

$$\theta_0 = 1,65;\ \dot{\theta_0} = 0\ k = 1;\ \beta = \frac{1}{2};\ A =;\ A = \frac{2}{3} \tag{2.6}$$

Ho ottenuto il moto caotico atteso. Ho realizzato due animazioni presso l'indirizzo `https://github.com/mario-ambrosino/NODE/tree/master/dataset/pendulum/GIF` e ho riportato alcuni fotogrammi significativi nelle immagini seguenti (cfr. **Figure 2.3** e **2.4**).

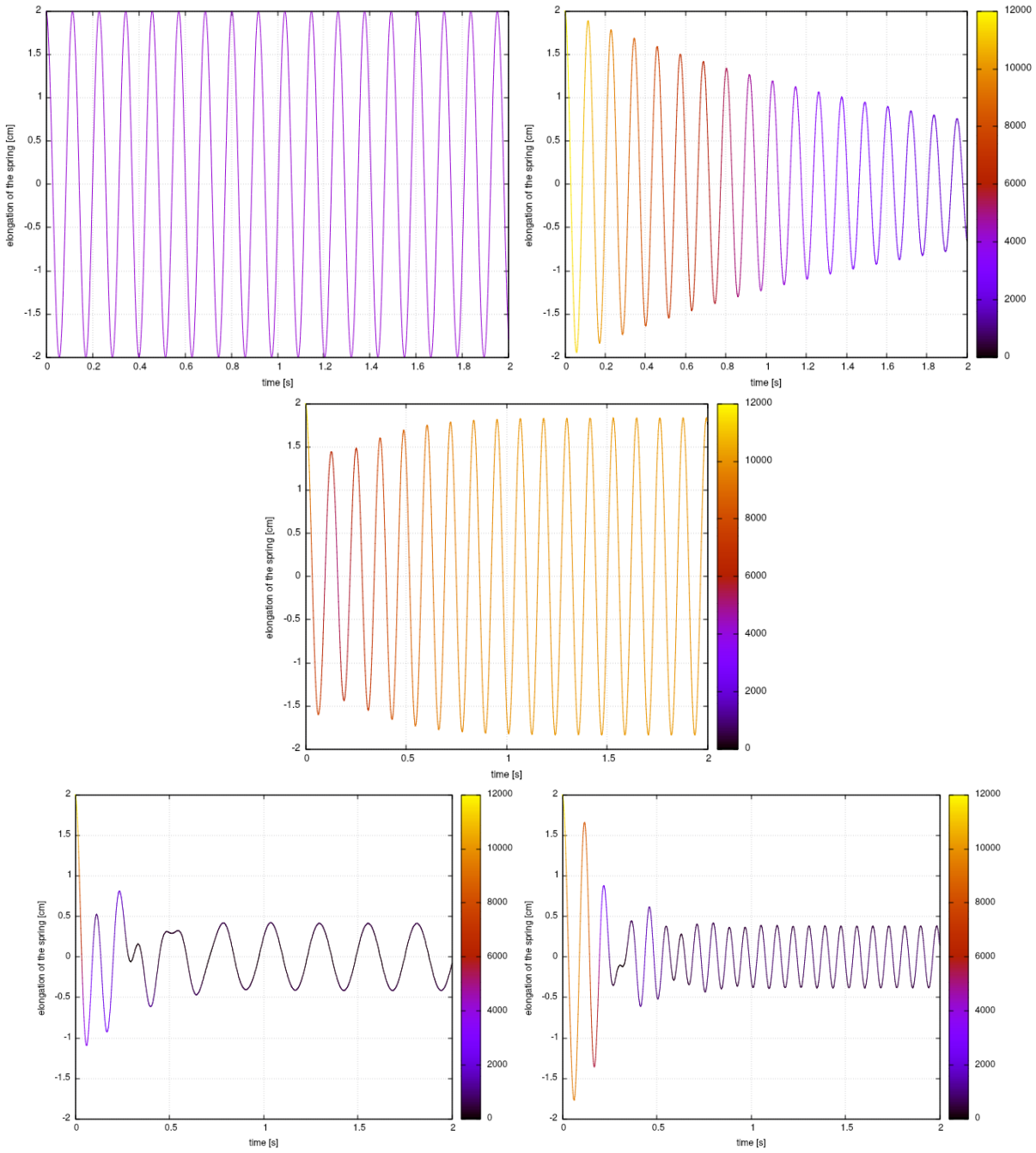


Figura 2.2: `forced_harmonic_oscillator.c`, In alto: evoluzione nel tempo dell'oscillatore armonico semplice, poi smorzato. Al centro: evoluzione nel tempo di un oscillatore armonico risonante. In basso: le condizioni prima e dopo la risonanza, al variare della frequenza della forzante del sistema.

Figura 2.3: `pendulum.c`, Ritratto di fase 2-D e spazio delle fasi in funzione del tempo per le frequenze $\omega_0 = 0.9, 1.07, 1.15$.

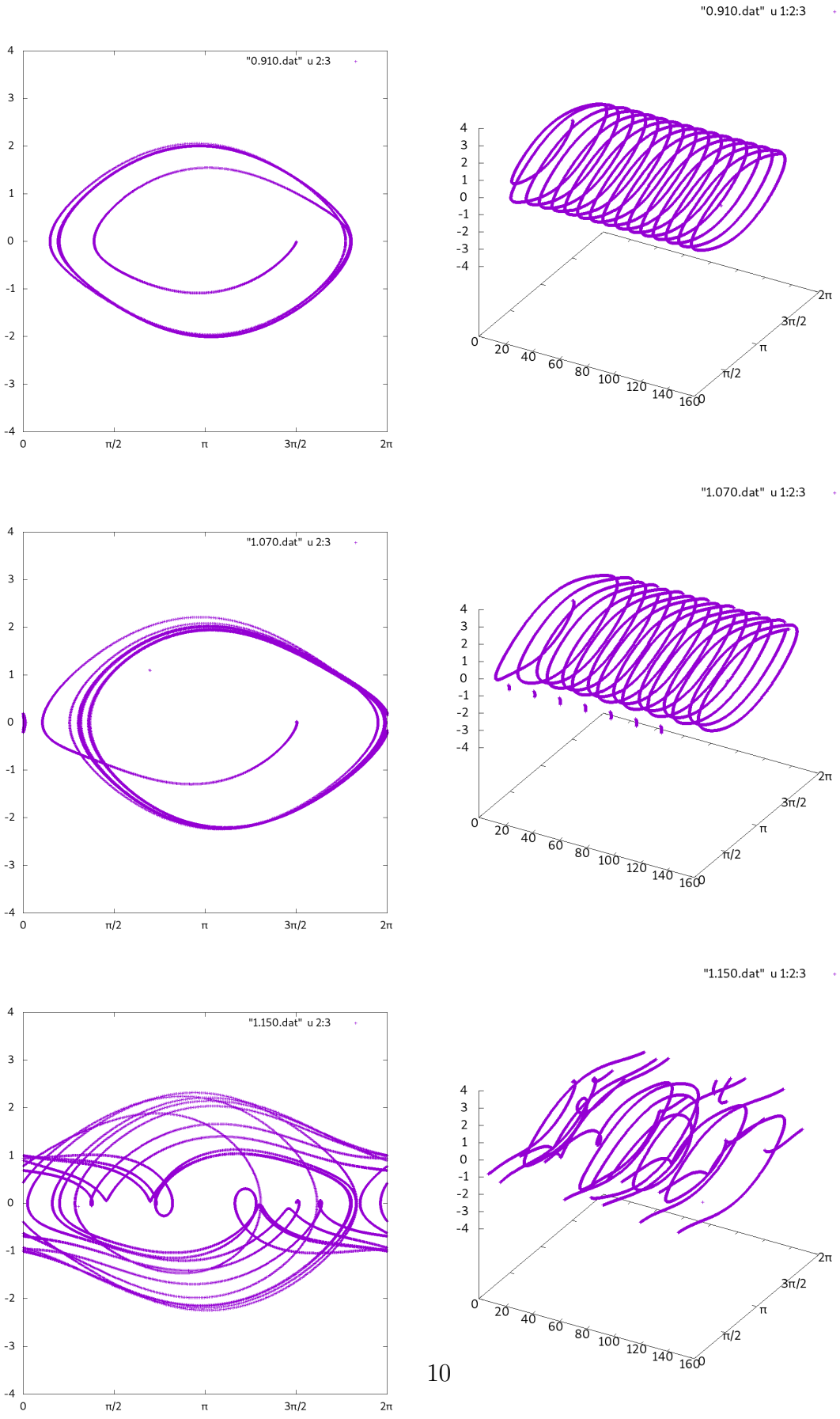
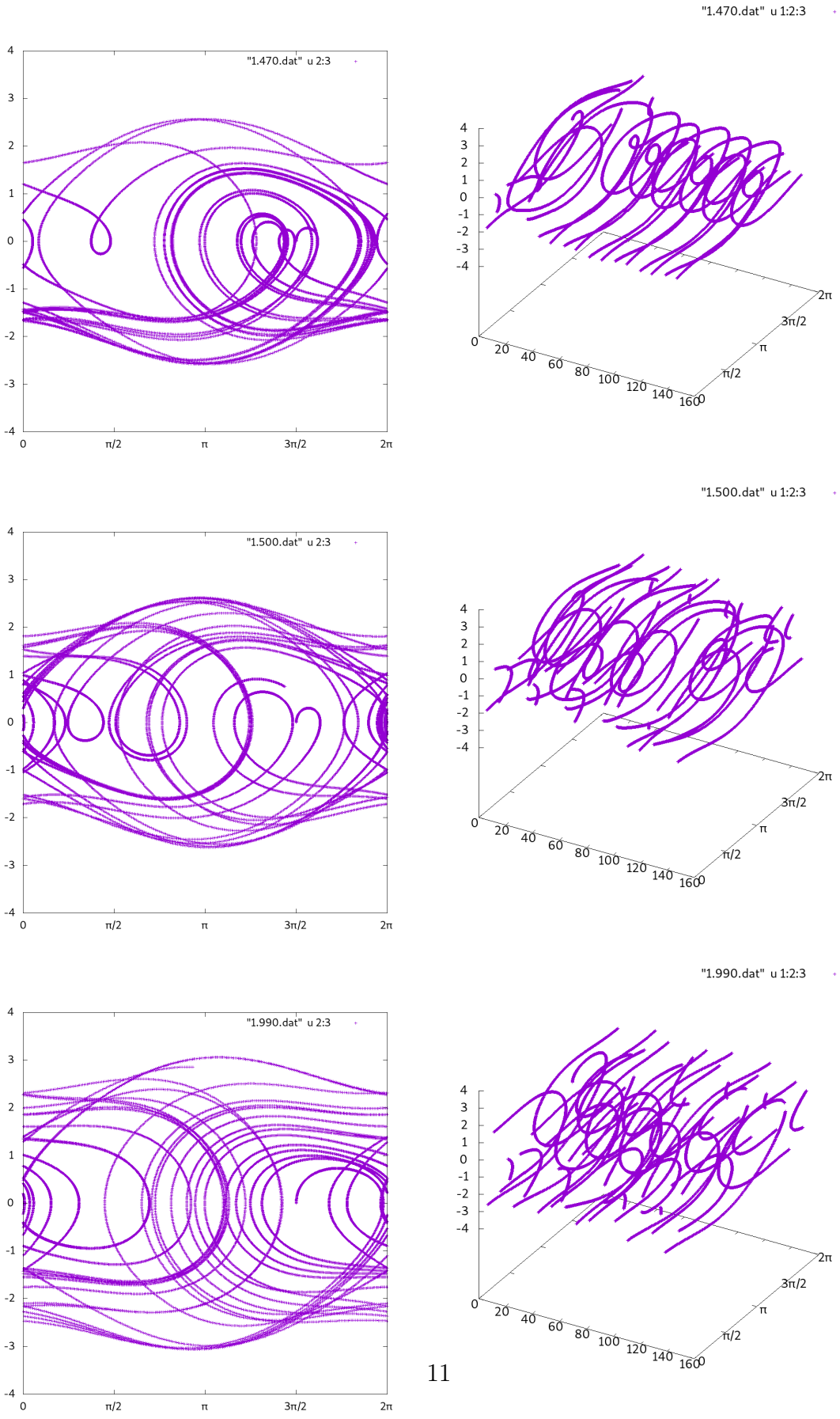


Figura 2.4: `pendulum.c`, Ritratto di fase 2-D e spazio delle fasi in funzione del tempo per le frequenze $\omega_0 = 1,47,\ 1.50,\ 1.99$.



Bibliografia

[1] L. Barone, E. Marinari, G. Organtini, F. Ricci-Tersenghi
Programmazione Scientifica
Pearson Education - (2006) - ISBN: 978-8-8719-2242-3

[2] Tao Pang
An Introduction to Computational Physics - Second Edition
Cambridge University Press - (2006) - ISBN: 978-0-521-53276-1