Quel temps fera-t-il la semaine prochaine ? Évolution d'un système chaotique simulé en virgule fixe

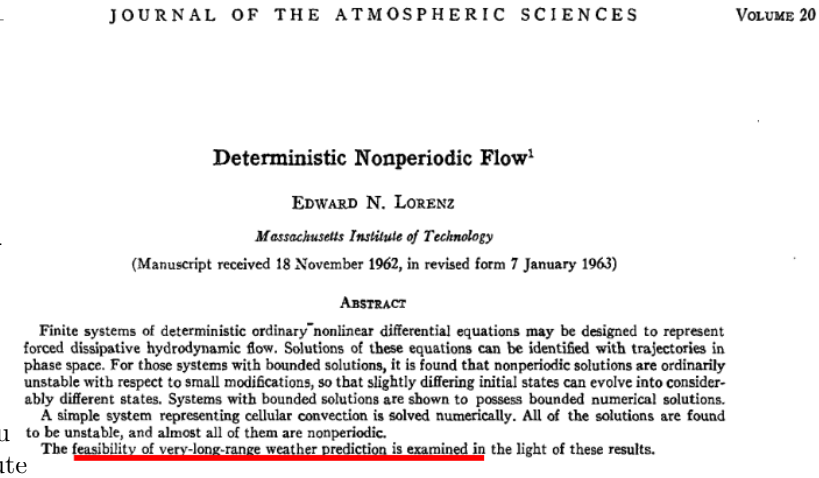
J.-M Friedt, 23 décembre 2022

enseignant-chercheur à l'Université de Franche-Comté à Besançon,

Dans le contexte du traitement du signal sur systèmes embarqués et en particulier sur microcontroleurs qui ne sont pas munis d'une unité de calcul en virgule flottante (FPU), nous reprenons une célèbre observation de 1963 sur l'évolution de systèmes dynamiques excessivement sensibles aux conditions initiales et donc qualifiés de chaotiques lors de simulations en représentation des nombres à virgules fixes.

René Descartes considérait au 17ème siècle que si les lois de la physique sont connues et les conditions initiales (position, vitesse des particules) de la modélisation d'un environnement sont identifiées, alors l'évolution du système est parfaitement prévisible. Cette vision dominante en occident pendant les deux siècles qui suivirent au cours du siècle des lumières et jusque dans les années 1950 [1] fut mise à mal par les physiciens de la seconde moitié du 20ème siècle qui gachèrent cet élégant déterminisme en introduisant les phénomènes non-linéaires, notamment nécessaires pour modéliser les fluides. Le plus grand fluide qui entoure cette planète est l'atmosphère, et la discussion de comptoir la plus intéressante reste encore la prévision du temps, déterminée par les mouvements de masses d'air induisant températures, précipitations et vents.

Edward Norton Lorenz (1917–2008) est le météorologue qui sera remémoré [2] pour avoir enterré le déterminisme de René Descartes en mettant en évidence la croissance exponentielle (et non polynomiale comme dans les lois considérées par Descartes) de l'erreur de la solution à un problème physique non-linéaire en fonction du temps avec l'erreur sur les conditions initiales. Couplée avec les relations d'incertitude temps-fréquence (ou position-vitesse) (ou temps-énergie) de Heisenberg, toute capacité de prévision de la météorologie à long terme est vaine, mais probablement la plus grand découverte de Lorenz est de concevoir que même si le temps (soleil, pluie, vent) à une date ultérieure lointaine ne peut être prévue, la solution reste bornée dans une gamme de valeurs plausibles. C'est la différence entre météorologie (quel temps fera-t-il ?) et climatologie (quelle sera la température moyenne au siècle prochain ?).



JOURNAL OF THE ATMOSPHERIC SCIENCES VOLUME 20

Deterministic Nonperiodic Flow[1]

EDWARD N. LORENZ

Massachusetts Institute of Technology

(Manuscript received 18 November 1962, in revised form 7 January 1963)

ABSTRACT

Finite systems of deterministic ordinary nonlinear differential equations may be designed to represent forced dissipative hydrodynamic flow. Solutions of these equations can be identified with trajectories in phase space. For those systems with bounded solutions, it is found that nonperiodic solutions are ordinarily unstable with respect to small modifications, so that slightly differing initial states can evolve into considerably different states. Systems with bounded solutions are shown to possess bounded numerical solutions.

A simple system representing cellular convection is solved numerically. All of the solutions are found to be unstable, and almost all of them are nonperiodic.

The feasibility of very-long-range weather prediction is examined in the light of these results.

Figure 1: Extrait de l'article original de E.N. Lorenz [2]

Tel que nous l'observons lors de nos séjours en arctique au Spitsberg, la prévision météorologique est toujours juste avec les modèles les plus fins actuels, mais la date à laquelle les évènements météorologiques surviennent est incertaine, et ce d'autant plus que la prévision est lointaine dans le futur : la pluie viendra bien après le beau temps, mais la date de transition est d'autant plus incertaine qu'elle est lointaine dans le futur. L'Arctique fut un enjeu majeur pendant la seconde guerre mondiale puisque les conditions météorologiques en Europe occidentale se déduisent des conditions météorologiques arctiques [3].

James Gleick [1] relate l'histoire de la découverte d'un système chaotique par E. Lorenz [4] par

"One day in the winter of 1961, wanting to examine one sequence at greater length, Lorenz took a shortcut. Instead of starting the whole run over, he started midway through. To give the machine its initial conditions, he typed the numbers straight from the earlier printout. Then he walked down the hall to get away from the noise and drink a cup of coffee. When he returned an hour later, he saw something unexpected, something that planted a seed for a new science.

This new run should have exactly duplicated the old. Lorenz had copied the numbers into the machine himself. The program had not changed. Yet as he stared at the new printout, Lorenz saw his weather diverging so rapidly from the pattern of the last run that, within just a few months, all resemblance had disappeared. He looked at one set of numbers, then back at the other. He might as well have chosen two

random weathers out of a hat. His first thought was that another vacuum tube had gone bad.

Suddenly he realized the truth. There had been no malfunction. The problem lay in the numbers he had typed. In the computer's memory, six decimal places were stored : .506127. On the printout, to save space, just three appeared : .506. Lorenz had entered the shorter, rounded-off numbers, assuming that the difference – one part in a thousand – was inconsequential.

[...]

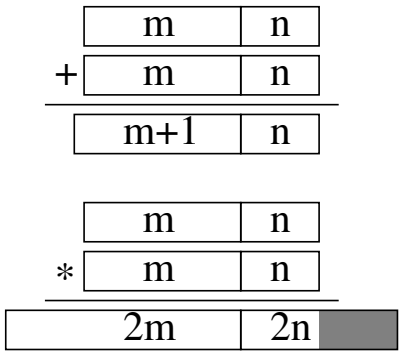
... in Lorenz's particular system of equations, small errors proved catastrophic. "

Ainsi donc, Edward Lorenz avait mis en évidence un système d'équations donc le comportement change drastiquement selon que l'on omette ou non les décimales de l'état représentant le système. Nos ordinateurs ne peuvent représenter le nombres qu'avec une précision finie : quelle conséquence a le chaos sur notre capacité de modélisation, et en particulier sur systèmes embarqués avec les contraintes de représentations des nombres à virgules ? À une époque ou l'"intelligence artificielle" et toutes les méthodes de traitement non-linéaires du signal sont à nouveau à la mode [5], il est peut être bon de se remémorer les limites imposées par la physique, aussi "intelligent" que nous voulions rendre l'ordinateur (à défaut de son programmeur).

1 Une balle qui tombe

Commençons cette démonstration par un système trivial qu'aurait prôné R. Descartes pour sa démonstration – une balle qui tombe dans le vide, parfaitement déterminisite et dont la solution ne dépend que faiblement de l'identification des conditions initiales. La simulation des équations différentielles régissant ce systèmes sont peu sensibles aux conditions de simulations et nous obtiendrons toujours le même résultat. Extrapolant les mêmes méthodes de simulation en virgule flottante ou virgule fixe avec un pas de temps de simulation variable, nous verrons que la modélisation de l'atmosphère sera considérablement modifiée ("météo") tout en restant bornée ("climat").

Arithmétique en virgule fixe



La représentation en virgule flottante représente chaque nombre sous forme scientifique de la forme mantisse (inférieure à 1 en valeur absolue) multipliée par un exposant, rendant les opérations arithmétiques complexes pour une unité arithmétique et logique (ALU) qui ne sait manipuler que des entiers, notamment pour aligner la position de la virgule lors de la somme. La représentation en virgule fixe bénéficie, par rapport à la représentation en virgule flottante, d'une arithmétique de la somme et la différence qui s'apparente à celle des entiers que l'ALU sait manipuler (figure de gauche, haut). Cependant, maintenir la position de la virgule lors des multiplications et divisions nécessite d'effectuer l'homothétie des arguments pour maintenir la position. Par exemple en décimal $10^{-2} \times 10^{-2} = 10^{-4}$ mais si seules 2 décimales sont significatives, les deux décimales ajoutées par le produit (10^{-3} et10^{-4}) doivent être éliminées pour maintenir la représentation sur deux décimales significatives. Cette opération est représentée par la zone grisée du produit en bas de la figure de gauche. Contrairement à l'arithmétique des entiers (`long`, `short` et `char` en C) qui est exacte car conservant tous les bits de poids faibles, ici n bits de poids faibles sont perdus lors de chaque multiplication.

Galileo Galilei définit la masse comme l'incapacité d'une corps à être mis en mouvement sous l'effet d'une force extérieure. L'accélération a que subit un corps détermine sa variation de vitesse v tel que la variation infinitésimale de vitesse dv pendant un temps infinitésimal dt est $\frac{dv}{dt} = a$ ou en d'autres termes $dv = a \cdot dt$ et entre deux instants séparés de dt la vitesse évolue de $v = v + dv$. De la même façon la vitesse est la dérivée avec le temps de la position, donc la balle qui tombe sous l'effet de la gravité voit sa position x évoluer par $\frac{dx}{dt} = v \Rightarrow dx = v \cdot dt$ et $x \leftarrow x + dx = x + v \cdot dt$. Avec v qui évolue tel que nous l'avons vu auparavant, il est évident d'effectuer la double intégration pour retrouver $x = \frac{1}{2} a \cdot t^2$ l'évolution de la position avec le temps. La résolution numérique du système

$$\begin{cases} \frac{dv}{dt} = a \Rightarrow v(t+dt) \leftarrow v(t) + a(t) \times dt \\ \frac{dx}{dt} = v \Rightarrow x(t+dt) \leftarrow x(t) + v(t) \times dt \end{cases}$$

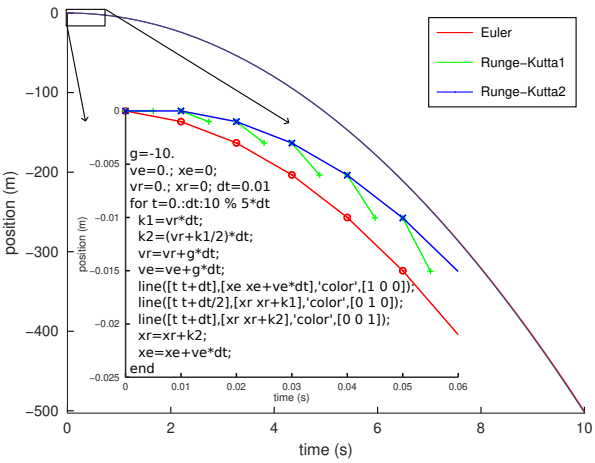
de la balle qui tombe en subissant l'accélération constante de la gravité de $a = 10$ m.s^{-2} sur la Terre, qui impacte l'évolution de la vitesse $v \leftarrow v + a \cdot dt$ qui impacte l'évolution de la position $x \leftarrow x + v \cdot dt$ entre chaque instant dt, se modélise trivialement selon la méthode d'Euler [6] que nous venons de décrire en représentation des variables à virgule flottante par

```
#include<stdio.h>
#include<stdlib.h> // atof
#include<math.h>   // pow : gcc ... -lm
int main(int argc, char** argv)
{float g=-10.,v=0.,x=0.;
 float dt=0.1,t,dx,dv;
 if (argc>1) dt=atof(argv[1]);
 for (t=0.;t<=10.+dt/2;t+=dt)
   {dv=g*dt; dx=v*dt;
    v=v+dv;  x=x+dx;
    printf("%f %f %f\n",t,x,.5*g*pow(t,2));
   }
}
```

L'infinitésimal étant très petit, nous pourrons arbitrairement choisir $dt = 0,1$ s, $dt = 0,01$ s ou $dt = 0,001$ s pour obtenir respectivement -505 m, -500,5 m, -500,05 m mais remonter à -499.21 m avec un pas de $dt = 0,1$ ms, et René Descartes ne verra que peut d'implication de l'intervalle de temps entre deux mesures, le système étant déterministe et l'approximation de la parabole par une série de segments de droites pas trop fausse en comparant avec la solution analytique $\frac{1}{2}at^2$ qui s'achève par $0,5 \times 10 \times 100 = 500$ m parcourus par la balle en 10 s. Cette formulation est la plus simple pour résoudre une équation différentielle mais la moins précise et le lecteur curieux d'améliorer le résultat pourra consulter [6, p.602] pour une description des méthodes de Runge-Kutta pour améliorer la précision de la solution.

Méthode de Runge-Kutta

La méthode d'Euler consistant à linéariser localement l'équation différentielle en considérant que le point suivant $x + dx$ se déduit uniquement de la droite liant le point courant par sa dérivée vers le point suivant $dx = v \times dt$ est un peu grossière et peut s'affiner en considérant des points intermédiaires aux temps $dt/2$, $dt/4$... Ces méthodes évoluées dites de Runge-Kutta, nommées d'après leurs concepteurs [7], améliorent l'exactitude et les chances de convergence de la solution tel qu'illustré sur la figure de droite, avec en insert le code GNU/Octave permettant de tracer ce graphique.



Nous avons déjà longuement expliqué [8] que la représentation en nombre à virgule flottante est inadéquate pour un système embarqué qui ne sait manipuler que des entiers, et qu'il est efficace d'utiliser la représentation en nombre à virgule fixe qui consiste à non plus conserver tous les bits de poids faible comme le fait l'artihmétique sur les entiers (avec potentiellement une croissance démesurée des bits de poids fort), mais de considérer que des bits de poids faibles ne sont que du bruit et qu'ils peuvent être éliminés lors de l'arithmétique sur les nombres à virgule fixe. Ainsi, en considérant que m bits représentent la partie entière d'un nombre et n bits représentent la partie fractionnaire (d'où la nomenclature introduite par Texas Instruments d'une représentation Qm.n), alors la somme de nombres en virgule fixe respecte l'arithmétique des entiers (la retenue se propagera de la partie fractionnaire vers la partie entière), mais la multiplication produit $2m$ bits de partie entière et $2n$ bits de partie fractionnaire dont la moitié est éliminée par décallage du résultat sur les entiers vers la droite de n bits. Ainsi, le programme que nous venons de voir en virgule flottante devient en virgule fixe en remplaçant les opérations arithmétiques "+" mais surtout "*" par `addfix()` et `mulfix()`, et en prenant soin d'effectuer l'homothétie sur les constantes que sont l'accélération et la date de fin de simulation :

```
#include<stdio.h>
#include "fixed.h"
int main()
{int g=(int)(-10.*SCALE),v=0.,x=0.;
 int dt=(int)(0.001*SCALE),t,dx,dv;
```

```
 for (t=0;t<=(10*SCALE);t+=dt)
   {dv=mulfix(g,dt);
    dx=mulfix(v,dt);
    v=addfix(v,dv);
    x=addfix(x,dx);
    printf("%d %d %d\n",t,SCALE,x);
   }
}
```

en exploitant les fonctions suivantes de calcul en virgule fixe qui conservent `SCALE` décimales (une version binaire avec un décallage au lieu de division par puissance de 10 est plus efficace mais moins pédagogique)

```
#include "fixed.h"
long addfix(long in1,long in2) {return(in1+in2);}

long mulfix(long in1,long in2)
{long long tmp;
 tmp=(long long)in1*(long long)in2;
 tmp/=SCALE;
 return((int)tmp);
}
```

qui utilise le fichier d'entête `fixed.h` définissant notamment la constante `SCALE` ou le nombre de décimales à conserver

```
long addfix(long ,long );
long mulfix(long ,long );
#define SCALE 100
```

Quelque soit l'intervalle de temps entre deux points de simluation, nous retrouvons un résultat proche de la solution analytique (Fig. 2 et l'hypothèse de Descartes est validée. Reproduisons maintenant la même expérience sur le système d'équations de Lorenz.

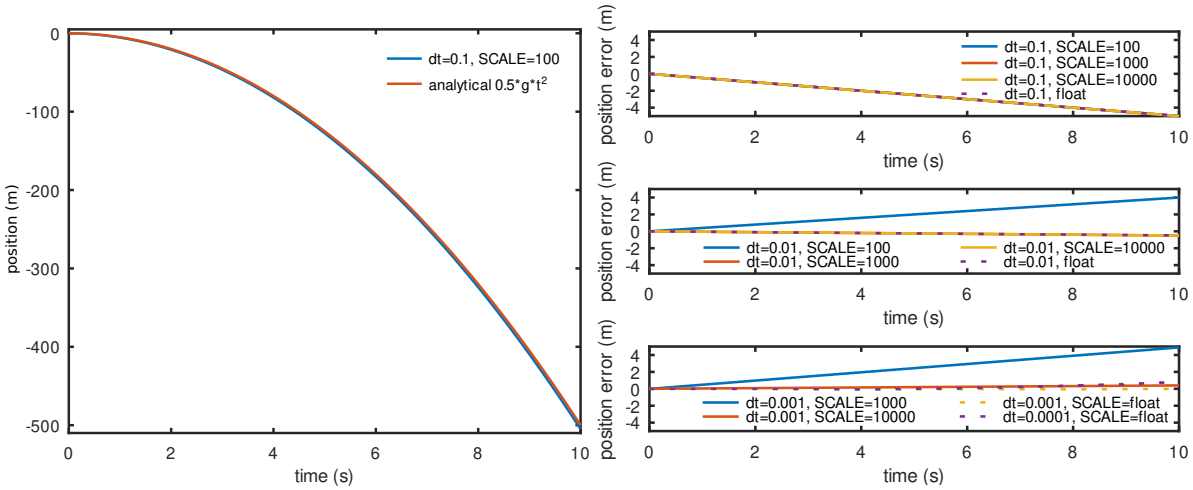


FIGURE 2 – Gauche : comparaison de la solution en virgule flottante et la solution en virgule fixe de la position d'une balle en chute libre dans le vide. Droite : erreur entre les diverses solutions numériques en virgule fixe et en virgule flottante en fonction du pas de temps dt et de la résolution déterminée par le nombre de chiffres significatifs après la virgule en virgule fixe, en prenant pour référence la solution analytique $1/2 \times a \times t^2$.

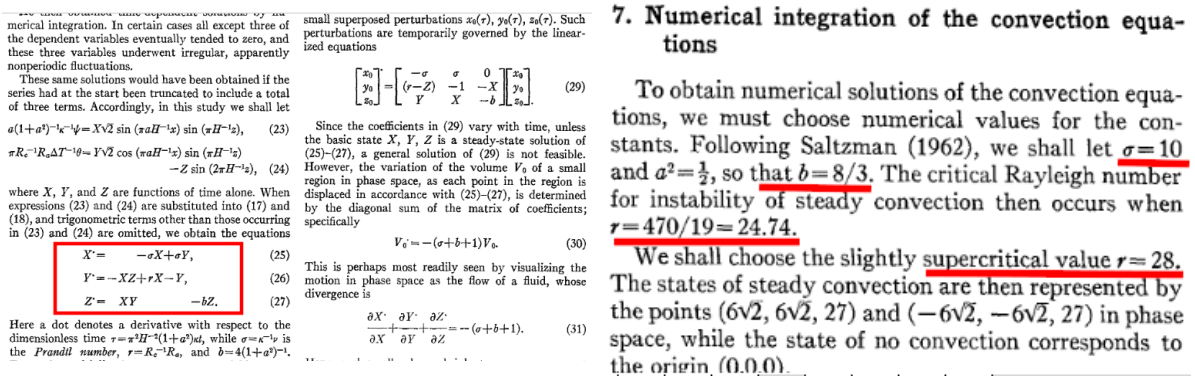
2 Un typhon ou un temps calme

Aux balbutiements de la résolution numérique de systèmes d'équations différentielles, notamment pour la modélisation des fluides et donc de l'atmosphère, alors que les ordinateurs possédaient moins de puissance de calcul que n'importe quel micrcontrôleur actuel, E.N. Lorenz poursuit un calcul en reprenant le résultat de la veille, mais en omettant d'insérer quelques décimales qu'il n'avait pas sauvegardé lors de l'impression de l'état de variables repésentant l'atmosphère selon trois grandeurs que nous nommerons arbitrairement (x, y, z) et que nous pourrons nous représenter par exemple comme x la température, y la

pression et z la vitesse du vent. Une modélisation simplifée de l'atmosphère se résume par un ensemble d'équation différentielles (Fig. 3) qui se résume [4] par

$$\begin{cases} \frac{dx}{dt} = -\sigma x + \sigma y \\ \frac{dy}{dt} = Rx - y - xz \\ \frac{dz}{dt} = -Bz + xy \end{cases}$$

avec les constantes $\sigma = 10$, $B = \frac{8}{3}$ and $R = 28$ (Fig. 3). L'équation de Lorenz est dite non-linéraire car fait intervenir des produits entre variables de la forme xz et xy, qui se traduit par un comportement chaotique défini comme une croissance exponentielle des erreurs sur les conditions initiales de la simulation ou le pas de temps de simulation. Ce comportement se distingue des systèmes linéaires que nous avons vu auparavant avec la balle qui tombe, dont les erreurs sur les conditions initiales croissent selon des lois polynomiales, donc beaucoup plus lentes (et donc déterministes). Cependant bien que chaotique, la solution de ce système reste bornée dans des gammes de solutions "raisonnables", par exemple avec des températures supportables et des vitesses de vent qui ne raseront pas des villes dans la majorité des conditions de simulations. La prévision du temps est considérée comme chaotique car il est très difficile (impossible) de connaître précisément les conditions de l'atmosphère à une date lointaine dans le futur, contrairement à la prévision du climat qui maintient les solutions bornées dans des solutions globalement reproductibles mais sans prétention d'annoncer quelle condition sera observée à quelle date.



merical integration. In certain cases all except three of the dependent variables eventually tended to zero, and these three variables underwent irregular, apparently nonperiodic fluctuations.

These same solutions would have been obtained if the series had at the start been truncated to include a total of three terms. Accordingly, in this study we shall let

$$a(1+a^2)^{-1}\kappa^{-1}\psi = X\sqrt{2}\sin(\pi a H^{-1}x)\sin(\pi H^{-1}z), \quad (23)$$

$$\pi R_c^{-1}R_a\Delta T^{-1}\theta = Y\sqrt{2}\cos(\pi a H^{-1}x)\sin(\pi H^{-1}z) - Z\sin(2\pi H^{-1}z), \quad (24)$$

where X, Y, and Z are functions of time alone. When expressions (23) and (24) are substituted into (17) and (18), and trigonometric terms other than those occurring in (23) and (24) are omitted, we obtain the equations

$$X' = -\sigma X + \sigma Y, \quad (25)$$

$$Y' = -XZ + rX - Y, \quad (26)$$

$$Z' = XY - bZ. \quad (27)$$

Here a dot denotes a derivative with respect to the dimensionless time $\tau = \pi^2 H^{-2}(1+a^2)\kappa t$, while $\sigma = \kappa^{-1}\nu$ is the *Prandtl number*, $r = R_c^{-1}R_a$, and $b = 4(1+a^2)^{-1}$.

small superposed perturbations $x_0(\tau)$, $y_0(\tau)$, $z_0(\tau)$. Such perturbations are temporarily governed by the linearized equations

$$\begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix}' = \begin{bmatrix} -\sigma & \sigma & 0 \\ (r-Z) & -1 & -X \\ Y & X & -b \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix}. \quad (29)$$

Since the coefficients in (29) vary with time, unless the basic state X, Y, Z is a steady-state solution of (25)–(27), a general solution of (29) is not feasible. However, the variation of the volume V_0 of a small region in phase space, as each point in the region is displaced in accordance with (25)–(27), is determined by the diagonal sum of the matrix of coefficients; specifically

$$V_0' = -(\sigma+b+1)V_0. \quad (30)$$

This is perhaps most readily seen by visualizing the motion in phase space as the flow of a fluid, whose divergence is

$$\frac{\partial X'}{\partial X} + \frac{\partial Y'}{\partial Y} + \frac{\partial Z'}{\partial Z} = -(\sigma+b+1). \quad (31)$$

7. Numerical integration of the convection equations

To obtain numerical solutions of the convection equations, we must choose numerical values for the constants. Following Saltzman (1962), we shall let $\sigma = 10$ and $a^2 = \frac{1}{2}$, so that $b = 8/3$. The critical Rayleigh number for instability of steady convection then occurs when $r = 470/19 = 24.74$.

We shall choose the slightly supercritical value $r = 28$. The states of steady convection are then represented by the points $(6\sqrt{2}, 6\sqrt{2}, 27)$ and $(-6\sqrt{2}, -6\sqrt{2}, 27)$ in phase space, while the state of no convection corresponds to the origin (0,0,0).

FIGURE 3 – Extraits de l'article original de E.N. Lorenz introduisant (gauche) son système d'équations couplées non-linéaires et la linéarization locale (Eq. 29), et (droite) les paramètres induisant un comportement chaotique.

En généralisant la méthode de résolution vue auparavant que l'évolution infinitésimale dx de chaque variable x évolue au cours de l'intevalle de temps infinitésimal dt pour une mise à jour $x \leftarrow x + dx$, l'opération est itérée sur les trois grandeurs (x, y, z). Dans un premier temps en représentation à virgule flottante et avec la condition initiale $(x, y, z) = (0, 1, 0., 0.)$, le programme de résolution s'écrit

```
#include <stdio.h>
int main()
{double x=0.1,y=0.,z=0.,sigma,R,B,dx,dy,dz;
 double dt=.001,t;
 sigma=10.; B=8./3.; R=470./19.;
 for (t=0;t<50.;t+=dt)
   {dx=sigma*(y-x)*dt;
    dy=(x*(R-z)-y)*dt;
    dz=(x*y-B*z)*dt;
    x=x+dx;
    y=y+dy;
    z=z+dz;
    printf("%f %f %f\n",x,y,z);
   }
}
```

et la première surprise survient lorsque nous traçons en fonction du temps x ou z pour des pas de simulation $dt = 10^{-2}$ ou $dt = 10^{-3}$. La forme des courbes est reproductible mais la transition d'un état à l'autre (haut ou bas de la courbe) intervient à des dates très différentes (Fig. 4). Toutefois, les solutions restent bornées dans des intervalles similaires.

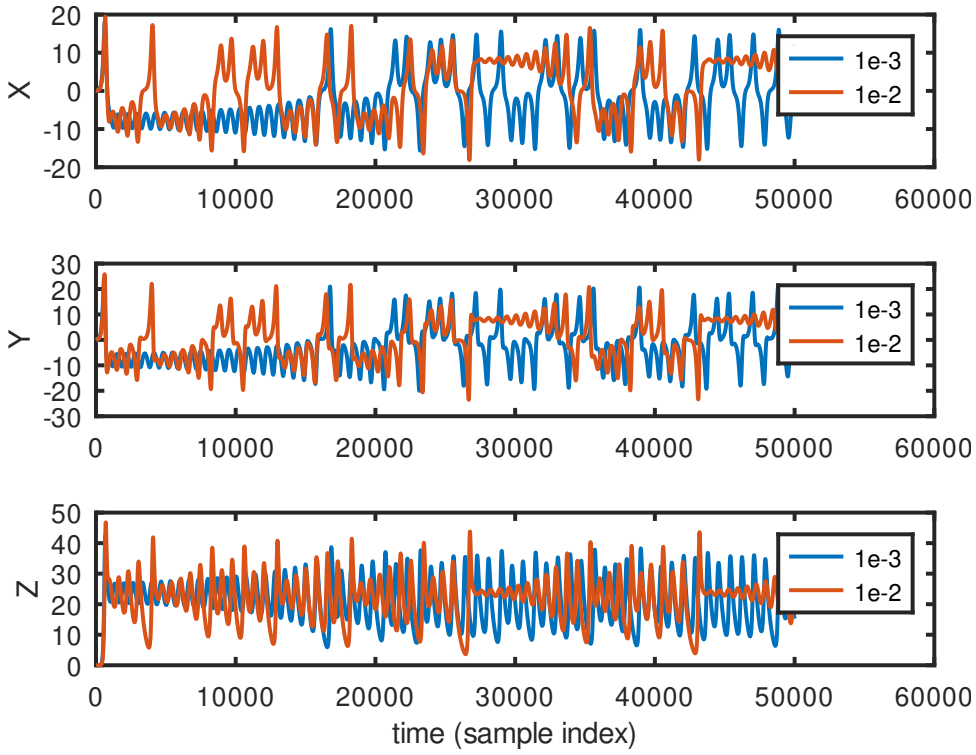


FIGURE 4 – Évolution avec le temps de la solution aux équations de Lorenz pour un pas de temps de $dt = 10^{-2}$ ou $dt = 10^{-3}$.

Par ailleurs, avec un pas de temps trop grossier, par exemple $dt = 10^{-1}$, la solution diverge pour ne plus être représentable par un nombre à virgule flottante et l'obtention de NaN ou *Not a Number*.

La transposition de la résolution par une représentation en virgule fixe s'exprime sous forme de

```
#include <stdio.h>
#include "fixed.h"

int main()
{int x=(int)(0.1*SCALE),y=0,z=0,sigma,R,B,dx,dy,dz;
 int dt=(int)(.01*SCALE),t;
 sigma=(10*SCALE);
 B=(int)((8./3.)*SCALE);
 R=(int)(470./19.*SCALE);
 for (t=0;t<(50*SCALE);t+=dt)
   {dx=mulfix(sigma,mulfix(addfix(y,-x),dt));
    dy=mulfix(addfix(mulfix(x,addfix(R,-z)),-y),dt);
    dz=mulfix(addfix(mulfix(x,y),-mulfix(B,z)),dt);
    x=addfix(x,dx);
    y=addfix(y,dy);
    z=addfix(z,dz);
    printf("%d %d %d\n",x,y,z);
   }
}
```

et en traçant cette fois l'évolution d'une variable, par exemple z, en fonction d'une autre variable, par exemple x, nous observons l'"attracteur étrange" qui exhibe la solution dans l'espace des phases (x, z), toujours avec un ensemble de solutions bornées mais se comportant très différemment selon que la pas de temps est $dt = 10^{-2}$ ou 10^{-3}, interdisant toute prévision de savoir si la solution se trouve sur l'aile droite ou gauche du papillon après un certain temps de simulation (Fig. 5).

L'exécution sur STM32F1 du code proposé à `https://github.com/jmfriedt/stm32/` dans le sous-répertoire `lorenz` indique une durée d'exécution de 1,25±0,05 seconde par itération d'une simulation de 50 secondes par pas de 10^{-3} en virgule fixe contre 1,38±0,05 s pour une version en virgule flottante en l'absence d'optimisation (option `-O0` par défaut de `gcc`) pour passer à 0,89±0,05 seconde par itération en

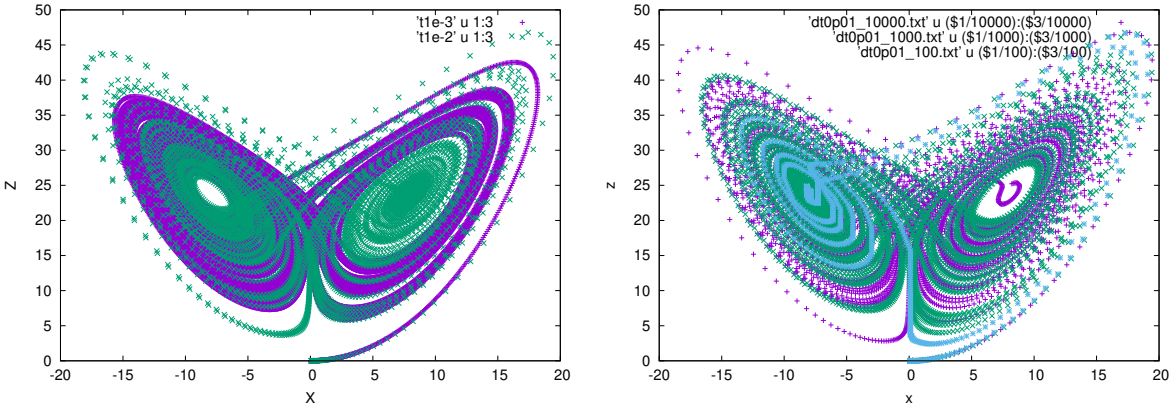


FIGURE 5 – Gauche : tracé d'une variable, z, en fonction d'une autre x, pour des conditions de simulation avec $dt = 10^{-2}$ ou avec $dt = 10^{-3}$. Droite : même tracé mais avec un pas de simulation fixe de 10^{-2} s et une résolution variable de 2, 3 ou 4 décimales de la représentation à virgule fixe.

virgule fixe lors d'une optimisation par `-O3` ou le classique `-Os`, qui n'amènent aucun bénéfice en virgule flottante. Ainsi, le bénéfice de la virgule fixe existe et devient significatif en laissant le compilateur optimiser l'exploitation des entiers par l'unité arithmétique et logique, pour gagner un facteur 1,5 sur le temps d'exécution, sachant que le facteur d'homothétie est une puissance de 10 et non une puissance de deux qui permettrait de remplacer les divisions entières par des décallages.

3 Conclusion

La résolution numérique de systèmes d'équations différentielles a été abordée dans le contexte de systèmes embarqués favorisant la représentation en nombre à virgule fixe sur les processeurs qui ne sont pas munis d'une unité matérielle de calcul en virgule flottante tel que le STM32F1. Alors que les systèmes physiques les plus courants respectant la linérité de leur comportement sont peu sensibles aux conditions initiales et conditions de simulation, les systèmes non-linéaires sont connus depuis les années 1960 et l'avènement de la thérorie du chaos pour être excessivement sensibles à ces conditions de simulation. Néanmoins, même s'il est difficile (impossible) de prédire à long terme la météo qu'il fera, les tendances climatiques qui sont modélisées par de tels systèmes restent bornées dans un ensemble cohérent de solutions, sans permettre d'identifier précisément à quelle date il fera beau. Et toute intelligence artificielle ne pourra pas déroger aux conditions imposées par la physique ...

Pour conclure, nous avons mentionné que la méthode d'Euler qui ne propage que la dérivée première pour prédire l'évolution du système est peu stable et que la méthode de Runge-Kutta présente de meilleures conditions de convergence en considérant la dérivée vers un point intermédiaire. Toutefois, notre code de résolution des équations de Lorenz par cette méthode présente bien entendu les mêmes sensibilités aux conditions de simulations dans le cas du système chaotique (Fig. 6) en ayant doublé le nombre d'opérations arithmétiques nécessaires à l'incrément du pas de temps.

Références

[1] J. Gleick, *Chaos : Making a New Science*, Random House UK (1997)

[2] E.N. Lorenz, *Deterministic nonperiodic flow*, J. of Atmospheric Sciences **20**(2) 130–141 (1963) at `https://journals.ametsoc.org/view/journals/atsc/20/2/1520-0469_1963_020_0130_dnf_2_0_co_2.xml`

[3] W. Dege, *War North of 80 – the last German Arctic weather station of World War II*, University of Calgary Press (2004)

[4] H.O. Peitgen, H Jürgens, D. Saupe, & M.J. Feigenbaum, *Chaos and fractals : new frontiers of science*, Springer New York (1992)

```c
#include <stdio.h>
int main()
{double x=0.1,y=0.,z=0.,sigma,R,B,t;
 double k1x,k1y,k1z,k2x,k2y,k2z,dt=.0001;
 sigma=10.; B=8./3.; R=470./19.;
 for (t=0;t<50.;t+=dt)
  {k1x=sigma*(y-x)*dt;   // intermediate step
   k1y=(x*(R-z)-y)*dt;
   k1z=(x*y-B*z)*dt;
   k2x=sigma*((y+k1y/2)-(x+k1x/2))*dt;
   k2y=((x+k1x/2)*(R-(z+k1z/2))-(y+k1y/2))*dt;
   k2z=((x+k1x/2)*(y+k1y/2)-B*(z+k1z/2))*dt;
   x=x+k2x;                // final increment
   y=y+k2y;
   z=z+k2z;
   printf("%f %f %f\n",x,y,z);
  }
}
```

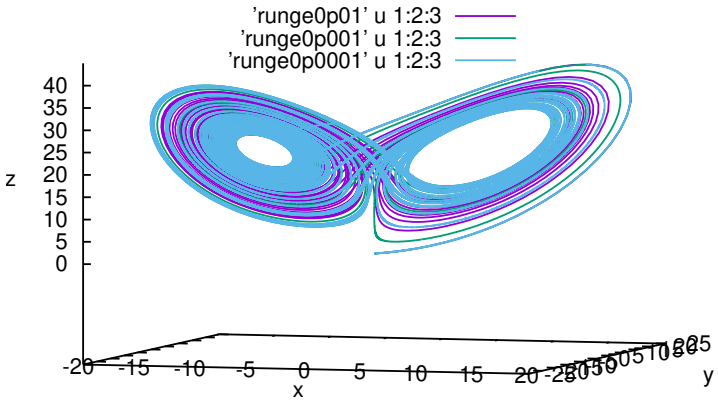


FIGURE 6 – Résolution des équations de Lorenz par la méthode de Runge-Kutta (gauche) et tracé des trois variables (x, y, z) en trois dimensions pour diverses valeurs du pas de temps de simulation de 10^{-2}, 10^{-3} et 10^{-4}.

[5] A. Sidder, *The AI Forecaster : Machine Learning Takes On Weather Prediction* (2022) à `https://eos.org/research-spotlights/the-ai-forecaster-machine-learning-takes-on-weather-prediction`

[6] Chap. 15 "Integration of Ordinary Differential Equations". dans W.H. Press *& al.*, *Numerical Recipes in Pascal – the Art of Scientific Computing*, Cambridge University Press (1989) ou Chap. 16 de *Numerical Recipes in C, 2nd Ed*, Cambridge University Press (1992)

[7] J.C. Butcher, *A history of Runge-Kutta methods.* Applied numerical mathematics **20**(3) 247–60 (1996) à `https://people.cs.vt.edu/~asandu/Public/Qual2011/DiffEqn/Butcher_1996_RK-history.pdf`

[8] J.-M Friedt, *Arithmétique sur divers systèmes embarqués aux ressources contraintes : les nombres à virgule fixe*, GNU/Linux Magazine France Hors Série **113** (Mars 2021)