BE : AUTOUR DE LA RÉSOLUTION D'ÉQUATIONS DIFFÉRENTIELLES

## Consignes générales :

- *Par groupe de 4 étudiants, vous devez rédiger un rapport dans lequel vous répondrez aux questions du sujet et plus globalement au problème demandé. La qualité de rédaction ainsi que tous les compléments que vous apporterez (réflexions, analyses des réponses, tests sur d'autres données, etc...) seront fortement pris en compte. Ce rapport devra être rédigé sous un traitement de texte (les pages manuscrites ne sont pas acceptées) et rendu sous la forme d'un fichier .pdf.*
- *Toutes les fonctions programmées le seront en Python et seront abondamment commentées. Le rendu du projet se fera sous la forme d'un fichier .zip contenant le rapport ainsi que l'ensemble des programmes et données nécessaire à l'exécution des programmes.*
- *Les fonctions programmées doivent l'être dans un fichier librairie ayant pour nom : **Ma322_BE_lib.py**. Dans ce fichier les différentes fonctions doivent être regroupés dans des blocs (commande : #%%) correspondants aux parties du sujet.*
- *Les applications numériques des fonctions du sujet seront dans un fichier ayant pour nom : **Ma322_BE_main.py**. Dans ce fichier les différents cas d'études doivent être regroupés dans des blocs séparés.*
- *Les différentes images devront être enregistrés dans un dossier ayant pour nom **Images**. Ce dossier doit être mis dans le même répertoire que vos fichiers pythons **Ma322_BE_lib.py** et **Ma322_BE_main.py**.*
- *Les noms des différents membres du groupe doivent apparaître en commentaire en début de fichier.*
- *Le non respect des consignes précédentes entraînera des points de pénalités pouvant être conséquents...*
- *Les questions d'analyses de résultats sont généralement ouvertes. La qualité de vos réflexions sera fortement prise en compte pour la notation de ces questions.*
- **Il est impératif de lire le sujet en entier.**
- **Ce sujet comporte peu de calculs théoriques mais de nombreuses fonctions Python à produire. Il est donc essentiel de vous partager les différentes tâches de codage.**

---

## Première partie

# Méthode de Verlet

Une difficulté avec les schémas numériques est leur stabilité dans le temps. En particulier lorsqu'un système dynamique est conservatif, le problème est de produire un schéma stable sur une longue durée. Par exemple, la question de la stabilité du système solaire d'étudier sur des périodes longues un système quasi-conservatif. Les algorithmes classiques d'ordre élevé, comme Runge-Kutta d'ordre 4, ont tendance à fournir une dynamique aux temps courts de bonne qualité mais non sur le long terme.

Dans tout cet exercice on se placera dans le cadre suivant. On décrira le mouvement d'un système dynamique conservatif de masse $m$ par une fonction de classe $\mathcal{C}^2(I, \mathbb{R}^3)$ que l'on notera :

$$X : \begin{array}{ccc} I & \longrightarrow & \mathbb{R}^3 \\ t & \mapsto & (x(t), y(t), z(t)) \end{array}$$

dont le mouvement est décrit par la relation fondamentale de la dynamique :

$$\ddot{X}(t) = \frac{1}{m} F(X(t))$$

où $F$ est l'expression vectorielle d'une force conservatrice et $\ddot{X}$ note la dérivée seconde par rapport au temps du vecteur position $X$.

On fixera $t_0, t_f$, respectivement le début du mouvement et la fin et $h := \dfrac{t_f - t_0}{N}$ avec $N$ un entier non nul fixé. On notera classiquement :

$$X_n := X(t_n)$$

et de même pour les autres fonctions intervenants.

1. Un défaut des méthodes classiques sur l'exemple de la méthode d'Euler.
   (a) Donner l'expression de la méthode d'Euler (explicite) du système dynamique précédent pour $t_0, t_f$fixé et $h := \dfrac{t_f - t_0}{N}$ avec $N$ un entier non nul fixé également, en fonction de $X, \dot{X}, F$ et $m$.

(b) Comme $F$ est conservatrice, il existe un potentiel $\Phi$ tel que :

$$-\frac{\partial \Phi}{\partial x_i}(X) = F_i(X)$$

où $x_i$ et $F_i$ sont les composantes $i$ des vecteurs $X$ et $F(X)$ respectivement. Alors en un instant $t$ l'énergie totale du système est donnée par :

$$\frac{m}{2}\left\|\dot{X}(t)\right\|^2 + \phi(X(t)) = E$$

avec $E$ constante réelle. Montrer que la méthode d'Euler ne conserve pas l'énergie totale du système. Pour cela :

i. on écrira en $t_n$ l'expression de l'énergie totale notée $E_n$en fonction des composantes de $\dot{X}_n, \Phi(X_n)$ et $m$,

ii. puis donner l'expression de $E_{n+1} - E_n$ en fonction de de $F(X_n)$,m et $h$ en utilisant le schéma d'Euler et en supposant lorsque $h$ est petit que l'approximation suivante est valide :

$$\frac{\phi(X_{n+1}) - \phi(X_n)}{h} = -\dot{X}_n^T F(X_n)$$

2. **La méthode de Verlet à deux pas (1967)**.

(a) En utilisant les développements de Taylor du vecteur position à l'ordre 3 aux instants $t+h$ et $t-h$,pour un $t$ fixé et $h$ petit, montrer que l'on peut poser le schéma itératif suivant :

$$X_{n+1} = 2X_n - X_{n-1} + \frac{h}{m}^2 F(X_n)$$

pour tout $N > n > 0$. De quel ordre est ce schéma ?

(b) Pour initialiser ce schéma il nous faut cependant connaître deux valeurs initiales : $X_{-1}$ et $X_0$. Justifier que l'on peut approximer $X_{-1}$ par la formule :

$$X_{-1} \simeq X_0 - h\dot{X}_0 + \frac{h^2}{2m}F(X_0)$$

On obtient alors lorsque l'on connaît l'expression de la force $F$, l'algorithme de Verlet :

$$\boxed{\begin{cases} X_0, \dot{X}_0 \text{ données} \\ X_{-1} = X_0 - h\dot{X}_0 + \dfrac{h^2}{2m}F(X_0) \\ X_{n+1} = 2X_n - X_{n-1} + \dfrac{h}{m}^2 F(X_n) \quad n \geq 0 \end{cases}}$$

3. **Application 1 (échauffement ) :** Le but de cette question est comparer le méthode de Runge-Kutta 4 et la méthode de Verlet pour prédire la position de planètes dans le système solaire à partir de positions et de vitesses initiales des planètes par exemple (Mars ,Terre, Soleil). Les données et leurs noms de variables Python seront :

- $LONG = 1.e11$
- $MASSE = 1.e24$
- $JOUR = 24. * 3600$
- $G = 6.67e-11/(LONG**3/MASSE/JOUR**2)$ : constante de gravitation en $m^3/kg/s^2$
- $mS = 2.0e30/MASSE$ : masse du soleil en kg .
- $mT = 6.0e24/MASSE$ : masse de la terre en kg.
- $mM = 6.418e23/MASSE$ masse de Mars en kg.
- $rT = 147.0e9/LONG$ : distance initiale Terre Soleil en m.
- $rM = 227939200e3/LONG$ : distance initiale Mars Soleil en m.
- $uT = 30.4e3 * JOUR/LONG$ : vitesse initiale de la Terre en m/s.
- $uM = 24.4e3 * JOUR/LONG$ : vitesse initiale de Mars en m/s.
- le Soleil : ( corps $i = 1$) sont : $XS = (0,0,0)^T$ et $VS = (0,0,0)^T$ .
- la Terre : ( corps $i = 2$) sont : $XT = (rT,0,0)^T$ et $VT = (0,uT,0)^T$ .
- la Mars : ( corps $i = 3$) sont : $XM = (rM,0,0)^T$ et $VM = (0,uM,0)^T$ .

(a) Écrire les équations différentielles du mouvement.

(b) Écrire de manière détaillée le problème de Cauchy associé au problème à 3 corps.

(c) Écrire une fonction Python permettant de calculer la force exercée par un corps $i$ sur un corps $j$.

(d) Écrire une fonction Python utilisant la méthode de Runge-Kutta 4 pour résoudre ce problème.

(e) Écrire une fonction Python utilisant la méthode de Verlet pour résoudre ce problème.

(f) Tracer la trajectoire de chaque corps.

(g) Tracer la norme de la différences des vecteurs positions entre les deux méthodes.

(h) Quelle est l'expression de l'énergie potentielle gravitationnelle ? Tracer alors l'énergie totale en fonction du temps pour les deux méthodes. Commenter ces derniers résultats.

4. **Application 2 :** le but de cette question et de généraliser la méthode de Verlet à n corps :

**Conditions initiales** : L'institut de mécanique céleste et de calcul des éphémérides (IMCCE) fournit les éphémérides des corps du système solaire. On utilise ce service pour obtenir les positions et vitesses initiales des corps.

Les éphémérides de l'IMCCE sont calculées avec des modèles de perturbation, qui utilisent des développements en série pour modéliser les différentes perturbations que les planètes exercent les unes sur les autres. Une méthode d'intégration numérique, similaire à celle que nous mettons en place ici, est utilisée par le Jet Propulsion Laboratory. Ces deux approches (théorie des perturbations et intégration numériques) sont complémentaires et contribuent toutes les deux à une connaissance précise des mouvements dans le système solaire. Le JPL a aussi un service d'éphéméride en ligne (Horizon).

Pour interroger le serveur d'éphémérides de l'IMCCE (Myriade), on peut utiliser le formulaire en ligne. Voici les paramètres à choisir :

— Target : choix d'une planète ou d'un satellite.

— Epoch : choix de la date initiale, du nombre de dates, et de l'intervalle de temps.

— Reference center : heliocenter.

— Advanced parameters : choisir le plan de référence écliptique et les coordonnées rectangulaires.

Obtenir les données sous forme d'un fichier texte. Voici un exemple, avec 5 dates espacées de 1 jour (réglage par défaut) :

```
# Miriade.ephemcc.results # Request : # targetType : Planet, targetNumber : 3, targetName : Earth, Diameter : 12756.27 km, Orbital period : 3.65256000E+02 d, Time_Scale : UTC, Planetary_Theory : INPOP13C, Coordinates : Ecliptic rectangular, frameType : Astrometric J2000, frameCentre : Heliocenter # Nb rows : 5 Target, Date, X (au), Y (au), Z (au), Xp (au/day), Yp (au/day), Zp (au/day), Observer distance (au)
Earth, 2023-04-17T03 :07 :00.00, -0.8978493720414, -0.4484569574624, 0.0000303993668, 0.0074065897847, -0.0154640808701, 0.0000003065817, 1.0036170278028
Earth, 2023-04-18T03 :07 :00.00, -0.8903108873215, -0.4638548033280, 0.0000306457623, 0.0076699810740, -0.0153308039893, 0.0000001915756, 1.0038997736790
Earth, 2023-04-19T03 :07 :00.00, -0.8825102381094, -0.4791169622811, 0.0000307941892, 0.0079308871467, -0.0151927171936, 0.0000001114769, 1.0041799564130
Earth, 2023-04-20T03 :07 :00.00, -0.8744500023943, -0.4942386608905, 0.0000308815060, 0.0081891249260, -0.0150498992633, 0.0000000696370, 1.0044573965878
Earth, 2023-04-21T03 :07 :00.00, -0.8661329320044, -0.5092152205493, 0.0000309462790, 0.0084445315063, -0.0149024595668, 0.0000000661485, 1.0047320029238
```

Le tableau comporte la date, les coordonnées héliocentriques en unité astronomique (AU), la distance au Soleil, et les composantes héliocentriques de la vitesse, en AU par jour. Voici comment extraire les positions et les vitesses du fichier :

```python
import numpy
data = numpy.loadtxt("miriade-earth.txt",skiprows=4,delimiter=",",usecols=(2,3,4,6,7,8)) Y_terre = data[0]
```

(a) Écrire les équations différentielles du mouvement pour n coprs.

(b) Écrire de manière détaillée le problème de Cauchy associé au problème à n corps.

(c) Écrire une fonction Python utilisant la méthode de Runge-Kutta 4 pour résoudre ce problème.

(d) Écrire une fonction Python utilisant la méthode de Verlet pour résoudre ce problème.

(e) Tracer la trajectoire de chaque corps.

(f) Tracer la norme de la différences des vecteurs positions entre les deux méthodes ainsi que l'énergie cinétique et potentielle.

## Deuxième partie

# Variations autours des équations de diffusions

Dans cet exercice nous allons utiliser l'algorithme de Runge-Kutta 4 afin de résoudre numériquement des variantes des équations de diffusion 2D. En guise d'application nous verrons l'effet de ces différentes équations sur des images.

La plus connues des équations de diffusion est l'équation de la chaleur. On rappelle que l'**équation de la chaleur 2D** est pour une fonction

$$u : [t_0, t_f] \times \Omega \longrightarrow \mathbb{R}$$

de classe $\mathcal{C}^2$ et où $\Omega$ est un domaine de $\mathbb{R}^2$ :

$$\begin{cases} \dfrac{\partial u}{\partial t}(t,x,y) = \gamma \Delta u(t,x,y) \\ u(0,x,y) = u_0(x,y) \end{cases}$$

avec $u_0$ une fonction connue.

## 1 Résolution de l'équation de la chaleur 2D

Dans cette section nous allons nous intéresser à la résolution numérique de l'équation de la chaleur suivante

$$(\mathcal{P}) : \begin{cases} \dfrac{\partial u}{\partial t}(t,x,y) = \Delta u(t,x,y) \\ u(0,x,y) = u_0(x,y) \\ \dfrac{\partial u}{\partial x}(t,x,0) = \dfrac{\partial u}{\partial x}(t,x,1) = 0 \\ \dfrac{\partial u}{\partial y}(t,0,y) = \dfrac{\partial u}{\partial y}(t,1,y) = 0 \end{cases}$$

où on a posé $\gamma = 1$.

On se placera dans le carré unité : $\Omega := [0,1]^2$. Fixons deux entiers $N \in \mathbb{N}^*$ et $M \in \mathbb{N}^*$ assez grand et notons $\Delta x = \dfrac{1}{M+1}$ et $\Delta y = \dfrac{1}{N+1}$. On a alors :

— un maillage du segment $\Omega$ par les points : $y_j := j\Delta y$, $x_i := i\Delta x$

— et on note alors : $u^t_{i,j} := u(t, x_i, y_j)$.

1. En appliquant l'opérateur d'approximation des différences finies montrer pour les dérivées d'ordre 2 de l'équation :

$$\Delta u(t, x_i, y_j) \simeq \left( \frac{u^t_{i+1,j} - 2u^t_{i,j} + u^t_{i-1,j}}{(\Delta x)^2} + \frac{u^t_{i,j+1} - 2u^t_{i,j} + u^t_{i,j-1}}{(\Delta y)^2} \right)$$

2. En déduire que pour tout $t \geq 0$, on a comme approximation de $(\mathcal{P})$ :

$$(\mathcal{P}_D) : \begin{cases} \dfrac{\partial u}{\partial t}(t,x_i,y_j) = \left( \dfrac{u^t_{i+1,j} - 2u^t_{i,j} + u^t_{i-1,j}}{(\Delta x)^2} + \dfrac{u^t_{i,j+1} - 2u^t_{i,j} + u^t_{i,j-1}}{(\Delta y)^2} \right) \\ u(0,x_i,y_j) = u_0(x_i,y_j) \\ \dfrac{u^t_{i+1,0} - u^t_{i,0}}{\Delta x} = \dfrac{u^t_{i+1,N+1} - u^t_{i,N+1}}{\Delta x} = 0 \\ \dfrac{u^t_{0,j+1} - u^t_{0,j}}{\Delta y} = \dfrac{u^t_{N+1,j+1} - u^t_{N+1,j}}{\Delta y} = 0 \end{cases}$$

3. Notons $U^t$ la matrice dont les coefficients sont donnés par :

$$\left(U^t\right)_{i,j} = u(t, x_i, y_j)$$

pour $(i,j) \in [\![1, M]\!] \times [\![1, N]\!]$ et notons la matrice « complétée » : $\tilde{U}^t$ la matrice dont les coefficients sont données par :

$$\left(\tilde{U}^t\right)_{i,j} = u(t, x_i, y_j)$$

pour $(i,j) \in [\![0, M+1]\!] \times [\![0, N+1]\!]$.

(a) Montrer qu'il existe deux matrices $D_1 \in \mathcal{M}_{M,M+2}(\mathbb{R})$ et $D_2 \in \mathcal{M}_{N,N+2}(\mathbb{R})$ tel que la première équation de $(\mathcal{P}_D)$ s'écrit matriciellement sous la forme :

$$\frac{dU^t}{dt} = D_1 \tilde{U}^t [:, 1:N] + \tilde{U}^t [1:M, :] D_2^T$$

où : $\tilde{U}^t [:, 1:N]$ note la sous-matrice de $\tilde{U}^t$ où l'on a conservé les colonnes de 1 à $N$. De même pour les lignes pour $\tilde{U}^t [1:M, :]$ .

(b) En déduire le problème de Cauchy associé à $(\mathcal{P}_D)$.

(c) Écrire l'algorithme (à la main) de Runge-Kutta 4 associé à ce problème de Cauchy.

4. Application. Pour les images on fixera $\Delta x = \Delta y = 1$. En guise d'application on prendra $U^0$ une image. Il est conseillé de commencer à faire vos algorithmes pour des images en nuance de gris (1 couches) puis de traiter ensuite le cas des images couleurs (3 couches).

(a) Rédiger une fonction Python **complétion(U)** qui prend en entrée une matrice image et qui renvoie la matrice $\tilde{U}$.

(b) Définir une fonction **Dmat(n)** qui prend en entrée un entier $n > 2$ et qui renvoient $D_1$ pour Dmat(M) et $D_2$ pour Dmat(N).

(c) Écrire un programme Python **RKimage(f,U0,t0,h,nbiter)** utilisant l'algorithme de Runge-Kutta 4 qui prend en entrée :

— $f$ la fonction donnée :

$$f\left(t, U^t\right) = D_1 \tilde{U}^t [:, 1:N] + \tilde{U}^t [1:M, :] D_2^T$$

— une image $U_0$,

— $t_0$ un réel positif, $h$ un pas et *nbiter* un nombre d'itération.

FIGURE 1 – Exemple de sortie : à gauche image modifiée, à droite image d'origine. On constate que l'équation donne un flou (gaussien).

## 2 Résolutions numériques de quelques variantes

Dans cette section nous allons « jouer » sur l'équation de la chaleur précédente en modifiant le terme de droite.

1. En reprenant les questions de la partie précédente rédiger un programme **RKimage_normgradconstant(f,U0,t0,h,nbiter)** qui résout numériquement le problème :

$$(\mathcal{P}) : \begin{cases} \dfrac{\partial u}{\partial t}(t,x,y) = \|\nabla u_0(x,y)\|_2 \\ u(0,x,y) = u_0(x,y) \\ \dfrac{\partial u}{\partial x}(t,x,0) = \dfrac{\partial u}{\partial x}(t,x,1) = 0 \\ \dfrac{\partial u}{\partial y}(t,0,y) = \dfrac{\partial u}{\partial y}(t,1,y) = 0 \end{cases}$$

FIGURE 2 – Exemple de sortie : à gauche image modifiée, à droite image d'origine. On constate que la diffusion du gradient de l'image originale accentue les contours de l'image.

2. En reprenant les questions de la partie précédente rédiger un programme **RKimage_normgrad(f,U0,t0,h,nbiter)** qui résout numériquement le problème :

$$(\mathcal{P}) : \begin{cases} \dfrac{\partial u}{\partial t}(t,x,y) = \|\nabla u(t,x,y)\|_2 \\ u(0,x,y) = u_0(x,y) \\ \dfrac{\partial u}{\partial x}(t,x,0) = \dfrac{\partial u}{\partial x}(t,x,1) = 0 \\ \dfrac{\partial u}{\partial y}(t,0,y) = \dfrac{\partial u}{\partial y}(t,1,y) = 0 \end{cases}$$

3. En reprenant les questions de la partie précédente rédiger un programme **RKimage_normlaplace(f,U0,t0,h,nbiter)** qui résout numériquement le problème :

$$(\mathcal{P}) : \begin{cases} \dfrac{\partial u}{\partial t}(t,x,y) = \sqrt{\left(\dfrac{\partial^2 u}{\partial x^2}(t,x,y)\right)^2 + \left(\dfrac{\partial^2 u}{\partial y^2}(t,x,y)\right)^2} \\ u(0,x,y) = u_0(x,y) \\ \dfrac{\partial u}{\partial x}(t,x,0) = \dfrac{\partial u}{\partial x}(t,x,1) = 0 \\ \dfrac{\partial u}{\partial y}(t,0,y) = \dfrac{\partial u}{\partial y}(t,1,y) = 0 \end{cases}$$

4. Tester d'autres fonctions à discrétiser à droite.
5. **Bonus/Concours :** les plus belles modifications (algorithmiques ! sans logiciel de traitement d'image) seront valorisées.

# 3 Diffusion anisotrope

Comme précédemment nous allons nous intéresser à l'équation de diffusion modifiée :

$$(\mathcal{P}_{aniso}) : \begin{cases} \dfrac{\partial u}{\partial t}(t,x,y) = \nabla.\left(c\left(\|\nabla u(t,x,y)\|^2\right)\nabla u(t,x,y)\right) \\ u(0,x,y) = u_0(x,y) \\ \dfrac{\partial u}{\partial x}(t,x,0) = \dfrac{\partial u}{\partial x}(t,x,1) = 0 \\ \dfrac{\partial u}{\partial y}(t,0,y) = \dfrac{\partial u}{\partial y}(t,1,y) = 0 \end{cases}$$

où $\nabla.(\bullet)$ note l'opérateur divergence et $c$ l'une des fonctions suivantes (à chacune étant à tester) :

$$c\left(s^2\right) := \frac{1}{1+\dfrac{s^2}{\lambda^2}} \qquad c\left(s^2\right) := e^{\left(-\frac{s^2}{\lambda^2}\right)}$$

avec $\lambda > 0$ fixé. Dans vos programmes vous testerez l'influence du paramètre $\lambda$.

1. Posons :

$$\begin{cases} c_{i,j}^t := c\left(\left(\dfrac{u_{i+1,j}^t - u_{i-1,j}^t}{2\Delta x} + \dfrac{u_{i,j+1}^t - u_{i,j-1}^t}{2\Delta y}\right)\right) \\ \left(c\dfrac{\partial u}{\partial x}\right)_{i+\frac{1}{2},j} \simeq \left(\dfrac{c_{i,j}^t + c_{i+1,j}^t}{2}\right)\left(\dfrac{u_{i+1,j}^t - u_{i,j}^t}{\Delta x}\right) := \phi_{i+\frac{1}{2},j} \\ \left(\dfrac{\partial}{\partial x}\left(c\dfrac{\partial u}{\partial x}\right)\right)_{i,j} \simeq \dfrac{\phi_{i+\frac{1}{2},j} - \phi_{i-\frac{1}{2},j}}{\Delta x} \end{cases}$$

   De même pour les dérivées partielles en $y$. Montrer que :

$$\frac{du_{i,j}^t}{dt} \simeq A_{x,i,j} + A_{y,i,j}$$

   où :

$$\begin{cases} A_{x,i,j} = \dfrac{1}{2\Delta x^2}\left(\left(c_{i,j} + c_{i-1,j}\right)u_{i-1,j} - \left(c_{i-1,j} + 2c_{i,j} + c_{i+1,j}\right)u_{i,j} + \left(c_{i,j} + c_{i+1,j}\right)u_{i+1,j}\right) \\ A_{y,i,j} = \dfrac{1}{2\Delta y^2}\left(\left(c_{i,j} + c_{i,j-1}\right)u_{i,j-1} - \left(c_{i,j-1} + 2c_{i,j} + c_{i,j+1}\right)u_{i,j} + \left(c_{i,j} + c_{i,j+1}\right)u_{i,j+1}\right) \end{cases}$$

2. En reprenant les questions des parties précédentes rédiger un programme **RKimage_anisotrope(f,U0,t0,h,nbiter)** qui résout numériquement le problème précédent avec la discrétisation précédemment démontrée. Vous testerez l'influence de la fonction $c$ ainsi que du paramètre $h$.

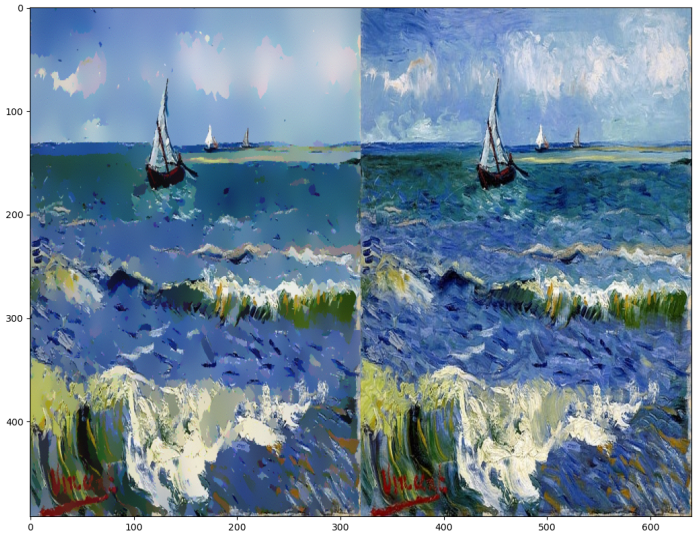


FIGURE 3 – Exemple de sortie : à gauche image modifiée, à droite image d'origine. On voit une diffusion anisotrope des couleurs. Les formes sont conservées. Cet algorithme est appliqué en particulier pour la détection de contour de tumeur en médecine, de différence de paysage sur des photos satellites,etc...