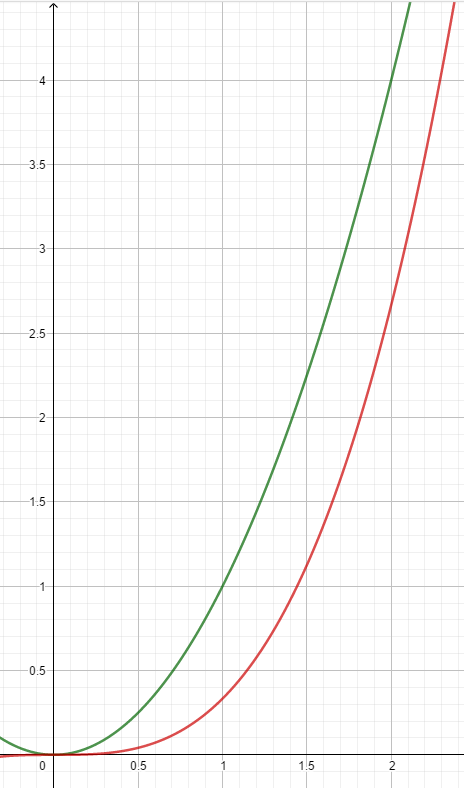
**Méthode d’Euler pour tracer la courbe de connaissant uniquement l’expression de**

Corrigé

***Exemple :*** Cas déjà résolu : Soit .

On connait déjà ce qui va nous servir pour contrôler notre approximation. Comment pourrait-on tracer une approximation de la courbe si on ne connaissait pas l’expression de , en sachant seulement que  ?



La dérivée de est ’

Donc le coefficient directeur de cette droite est

*Réponse :*

On utilise un algorithme qui trace point par point une approximation de la courbe .

Principe de l’algorithme :

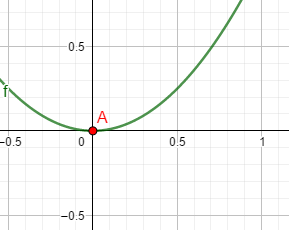
On se fixe « un pas » c’est-à-dire un réel strictement positif assez proche de zéro.

# Premier exemple avec peu de points

On choisit

## Tracé à la main de l’approximation de la courbe de la fonction *f*

On sait que donc la courbe passe par

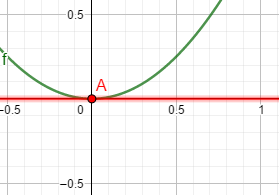


* On sait que est le nombre dérivé de calculé en .

On a .

Ce nombre dérivé est le coefficient directeur de la tangente à la courbe au point d’abscisse .

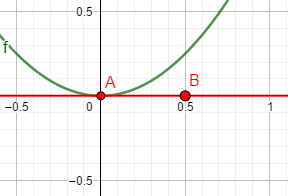
On trace cette tangente de coefficient directeur 0.



La tangente est a*pproximativement confondue* avec la courbe

Donc provisoirement la tangente nous sert à tracer l’approximation de la courbe

Comme le pas choisi est , on trace la tangente jusqu’à son point d’abscisse .

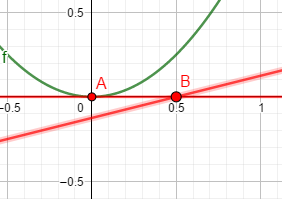


* est le nombre dérivé de calculé en

On a

Ce nombre dérivé nous donne la direction vers laquelle tracer notre approximation de courbe .

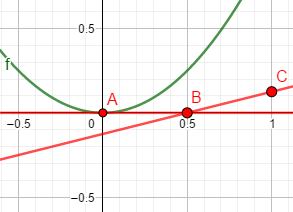
On trace la droite passant par ayant comme coefficient directeur



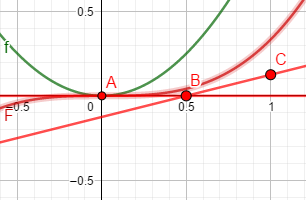
La droite est a*pproximativement confondue* avec la courbe

Donc provisoirement la droite nous sert à tracer l’approximation de la courbe

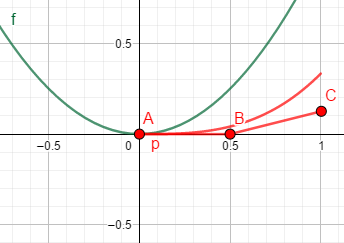
Comme le pas choisi est , on trace la droite jusqu’à son point d’abscisse .



On peut comparer la ligne brisée ABC qui est *l’approximation de la courbe* avec la véritable courbe puisqu’ici nous connaissons l’expression de



La différence entre la courbe et son approximation est assez importante. Cela est dû au fait que le pas choisi est assez grand.



## Algorithme pour calculer les coordonnées successivement des points

def fprime(x):

Principe :

    return x\*\*2

Droite de coefficient droite de coefficient directeur

def euler():

    y = 0

    x = 0

    print(x, y)

    for i in range(2):

        y = y + 0.5\*fprime(x)

        x = x + 0.5

        print(x, y)

On exécute la fonction euler()

écart en = écart en  **coefficient directeur**

Résultat :

0 0 (coordonnées de A)

0.5 0.0 (coordonnées de B)

1.0 0.125 (coordonnées de C)

# Augmentation du nombre de points

**Exercice 1 Algorithme et tableur**

On modifie le programme précédent pour qu’il affiche les coordonnées des points de *l’approximation* de d’abscisses (ce qui correspond à un pas ).

1. Écrivez ci-dessous le programme modifié.

def fprime(x):

    return x\*\*2

def euler():

    y = 0

    x = 0

    print(x, y)

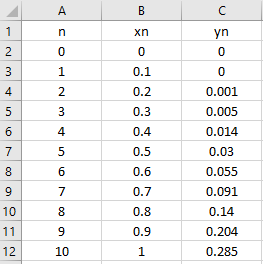
    for i in range(10):

        y = y + 0.1\*fprime(x)

        x = x + 0.1

        print(x, y)

1. En faisant fonctionner le programme modifié, complétez le tableau ci-dessous :



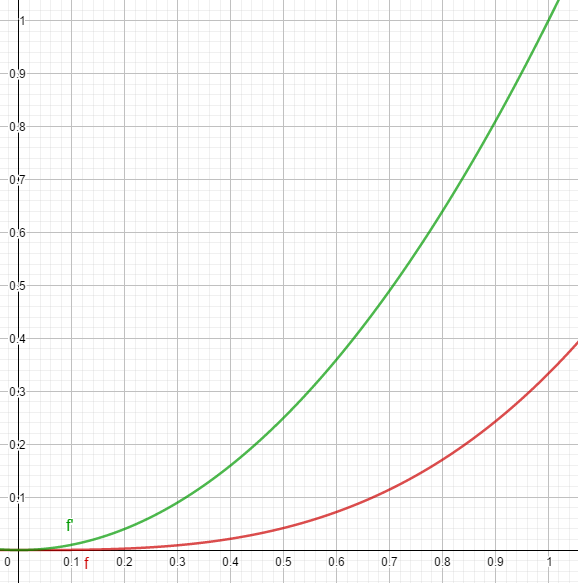
1. Si on voulait faire ces calculs dans une feuille de tableur, quelles formules devrait-on écrire dans les cellules B3 et C3 de manière à pouvoir les recopier vers le bas ?

En B3 on a la formule = B2 + 0.1 et en C3 on a la formule = C2 + 0.1 \* B2^2

1. Donnez l’expression des suites et des coordonnées des points de la ligne brisée qui approche la courbe

et

1. Placez sur le graphique suivant les points correspondant aux coordonnées du tableau précédent.



1. Comparez la précision de l’approximation de obtenue avec le pas et celle obtenue avec le pas .

Avec un pas de l’approximation est beaucoup plus proche de la vraie courbe

1. Que faudrait-il faire si on voulait que l’approximation se rapproche encore davantage de la véritable courbe  ?

Il faudrait encore diminuer le pas .

**On retient :**

***Méthode d’Euler :***

**Exercice 2 Équation différentielle**

1. On considère la fonction qui est la solution de l’équation différentielle

et qui vérifie la condition initiale .

Dans cette question, on suppose qu’on ne sait pas trouver l’expression exacte de .

On va utiliser la méthode d’Euler pour tracer approximativement la courbe de la fonction .

On a :

Exprimer uniquement en fonction de et de .

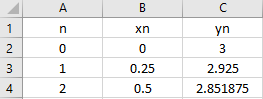
1. Donnez en fonction de l’expression des suites et des coordonnées des points de la ligne brisée qui approche la courbe .

et

1. On choisit le pas . Donnez l’expression des suites et des coordonnées des points de la ligne brisée qui approche la courbe

et

1. On a utilisé un tableur pour calculer les coordonnées de la ligne brisée qui approche la courbe  :



Quelles formules a-t-on écrit dans les cellules et de manière à pouvoir les recopier vers le bas ?

En B3 on a la formule = B2 + 0.25 et en C3 on a la formule = 0.975\*C2

1. Quelle est la plus petite valeur de pour laquelle on a d’après ce tableau ?

On fait la table de la suite sur la calculatrice ; On trouve ce qui donne **.**

1. Résoudre de façon exacte l’équation différentielle

et déterminer l’expression de la fonction qui vérifie la condition initiale .

. équivaut à donc d’où

1. Résoudre de façon exacte l’inéquation .

donc donc donc

1. La valeur de trouvée à la question 5 par la méthode d’Euler (qui est une méthode approximative) vous parait-elle bonne ?

et l’approximation avait donné . Donc l’approximation parait bonne.

**Exercice 3 Équation différentielle dont *on ne connait pas* l’expression de la solution**

Pour construire une approximation de la courbe représentative d’une fonction en utilisant la méthode d’Euler, on construit une suite de points tels que

avec réel positif et

) en utilisant l’approximation affine ***: .***

On considère l’équation différentielle :

avec .

* Ne connaissant pas la forme générale de la solution de cette équation, utilisez la méthode d’Euler[[1]](#footnote-1) avec pour programmer en Python le calcul des coordonnées permettant de tracer une approximation de la courbe représentative de cette solution sur l’intervalle[0 ; 2]. Vous écrirez votre programme ci-dessous.

On a :

On exprime uniquement en fonction de et de .

donc

et

D’où le programme Python :

def programme():

    y = 0

    x = 0

    print(x, y)

    for i in range(20):

        y = y + 0.4 - 0.1\*(y\*\*2)

        x = x + 0.1

        print(x, y)

programme()

Résultat :

0 0

0.1 0.4

0.2 0.784

0.30000000000000004 1.1225344000000002

0.4 1.396526052081664

0.5 1.601497550667384

0.6 1.745018110188021

0.7 1.840509289699604

0.7999999999999999 1.90176184515255

0.8999999999999999 1.940092033584747

0.9999999999999999 1.963696323706847

1.0999999999999999 1.9780859985328685

1.2 1.986803576773691

1.3 1.992064731505618

1.4000000000000001 1.995232542054763

1.5000000000000002 1.9971372523673316

1.6000000000000003 1.9982815318879983

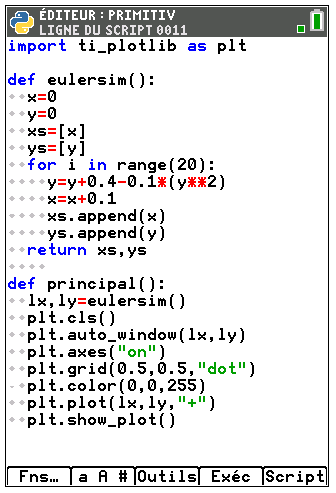
1.7000000000000004 1.9989686238195339

1.8000000000000005 1.9993810679180377

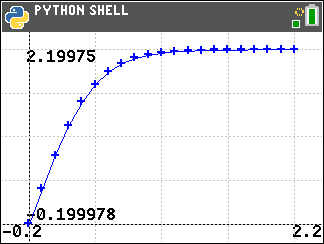
1.9000000000000006 1.9996286024431305

2.0000000000000004 1.9997771476722637

Sur la calculatrice TI 83 (système d'exploitation au moins version 5.5):



Après exécution, on obtient :



1. Léonard Euler : Mathématicien suisse (1707 – 1783) [↑](#footnote-ref-1)