TP8 Séries de Fourier

In [1]:
```
from math import *
import numpy as np
import matplotlib.pyplot as plt
import numpy.linalg as la
```

Partie 1 : Manipulations élémentaires

Soit f une fonction 1-périodique et intégrable. Pour $n \in \mathbb{Z}$, on note $c_n(f)$ son coefficient de Fourier d'ordre~n~:

$$c_n(f) = \int_0^1 f(t) e^{-2i\pi nt} \mathrm{d}t. \tag{1}$$

La méthode des rectangles (qui coïncide ici avec celle des trapèzes) fournit l'approximation suivante

$$c_n^{(K)}(f) = \frac{1}{K} \sum_{k=0}^{K-1} f\left(\frac{k}{K}\right) e^{-2i\pi \frac{kn}{K}}. \tag{2}$$

On choisira pour K un entier pair.

Calcul des coefficients de Fourier approchés.

In [2]:
```
def fourier(f,k):
    # renvoie les coefficients de Fourier d'indice 0 à 2k-1 de f (K = 2k)
    x = np.linspace(0,1,2*k+1)[0:2*k] # subdivision de [0,1] en 2k sous-intervall
    c = np.zeros(2*k, dtype=complex)
    C = np.zeros(2*k, dtype=complex)
    for n in range(K):
        ee = np.exp(-2*1j*pi*x*n)
        c[n] = np.vdot(f(x),ee)/(2*k)
    # réarrangement des termes pour renvoyer les fréquences entre -k et k-1
    C[0:k] = c[k:2*k] # fréquences de -k à -1
    C[k:] = c[0:k]    # fréquences de 0 à k-1
    return C
```

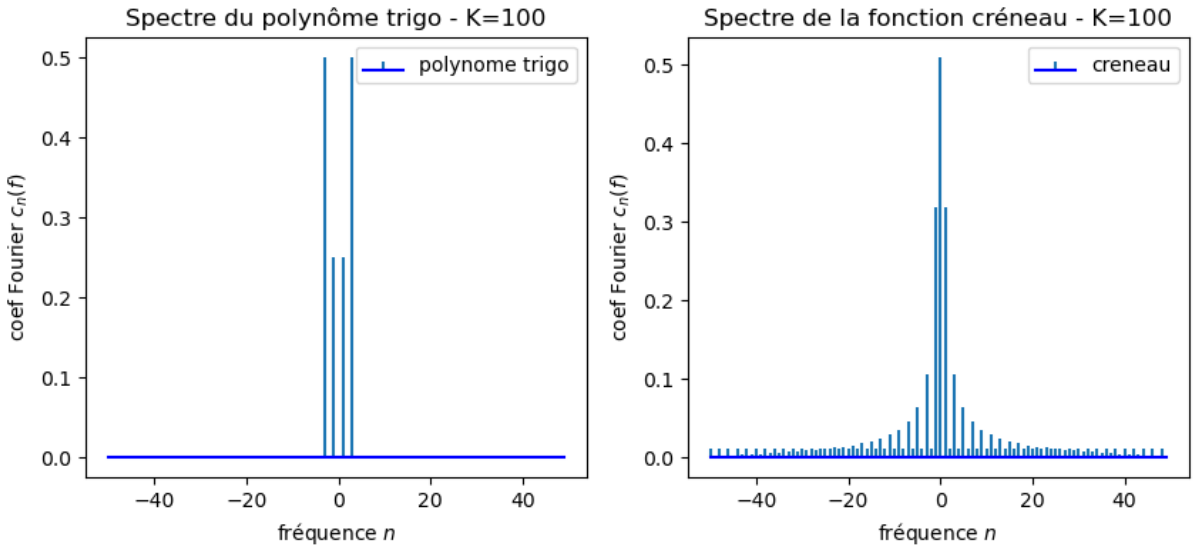
In [3]:
```
# Polynôme trigonométrique
def poly(x):
    return np.cos(6*pi*x)+ 0.5*np.sin(2*pi*x)


# Fonction créneau
def cren(x):
    return np.array((x<=3./4)*(x>=1./4),dtype=float)

K = 100
k = floor(K/2)
Fpoly = fourier(poly,k)
Fcren = fourier(cren,k)

plt.rcParams['figure.figsize'] = (10, 4)
plt.subplot(1,2,1)
plt.stem(range(-k,k),abs(Fpoly),label='polynome trigo', markerfmt=" ", basefmt="-
plt.title('Spectre du polynôme trigo - K=100')
plt.xlabel('fréquence $n$')
```

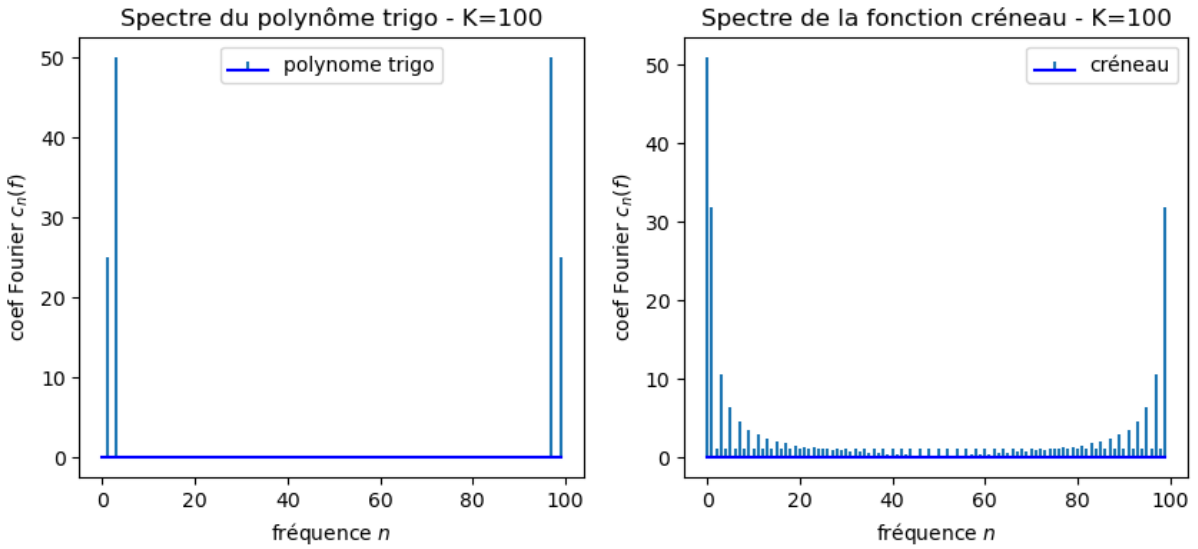
```
plt.ylabel('coef Fourier $c_n(f)$')
plt.legend()
plt.subplot(1,2,2)
plt.stem(range(-k,k),abs(Fcren),label='creneau', markerfmt=" ", basefmt="-b")
plt.title('Spectre de la fonction créneau - K=100')
plt.xlabel('fréquence $n$')
plt.ylabel('coef Fourier $c_n(f)$')
plt.legend()
plt.show()
```



In [4]:

```
K = 100
k = floor(K/2)
from numpy.fft import fft, ifft

x = np.linspace(0,1,K+1)[0:K]
plt.rcParams['figure.figsize'] = (10, 4)
plt.subplot(1,2,1)
plt.stem(range(0,2*k),abs(fft(poly(x),K)),label='polynome trigo', markerfmt=" ",
plt.title('Spectre du polynôme trigo - K=100')
plt.xlabel('fréquence $n$')
plt.ylabel('coef Fourier $c_n(f)$')
plt.legend()
plt.subplot(1,2,2)
plt.stem(range(0,2*k),abs(fft(cren(x),K)),label='créneau', markerfmt=" ", basefmt
plt.title('Spectre de la fonction créneau - K=100')
plt.xlabel('fréquence $n$')
plt.ylabel('coef Fourier $c_n(f)$')
plt.legend()
plt.show()
```



On note que les coefficients de Fourier sont calculés d'abord pour les fréquences entre 0 et $k-1$, puis pour les fréquences entre $-k$ et -1. On note également que fft ne normalise pas les coefficients, la valeur calculée est K fois la valeur calculée par la fonction fourier.

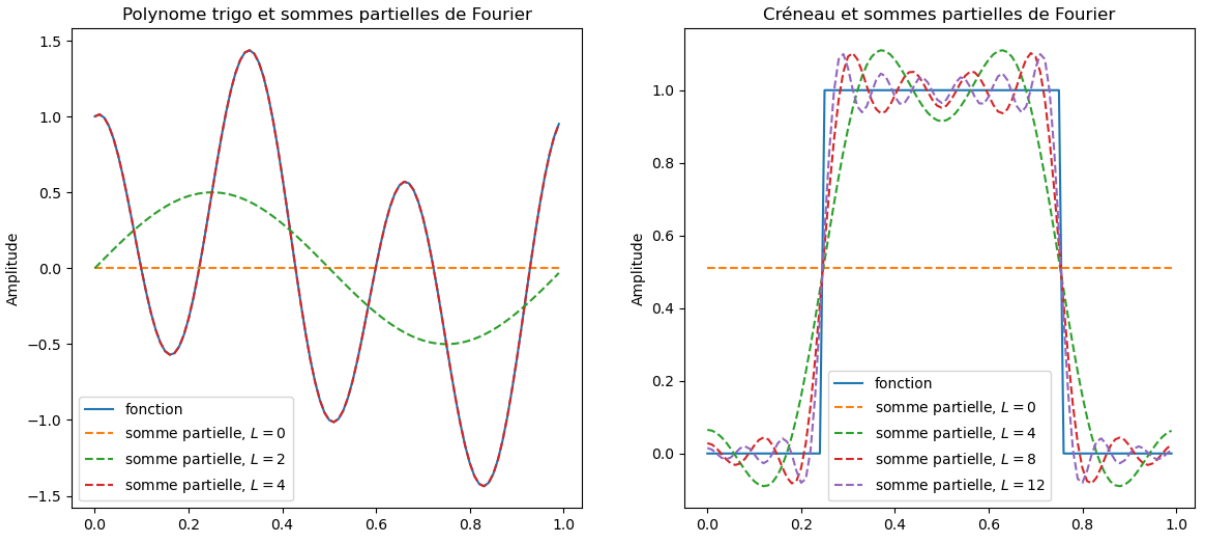
Calcul de la somme partielle de Fourier

In [5]:
```
def expo(x,n): # monôme trigonométrique
    return np.exp(2*1j*pi*x*n)

def partielle(f,K,L):
# somme partielle de Fourier de la fonction f, K = nb de points de discrétisation
    k = int(K/2)
    F = fourier(f,k)
    y = np.ones(2*k, dtype=complex)*F[k]
    x = np.linspace(0,1,K+1)[0:K]
    for l in range(1,L):
        y += F[k+l]*expo(x,l)+F[k-l]*expo(x,-l)
    return(y)

K = 100
plt.figure(figsize = (14, 6))
x = np.linspace(0,1,K+1)[0:K]
plt.subplot(121)
plt.plot(x, poly(x),label='fonction')
plt.ylabel('Amplitude')
for l in range(0,5,2):
    y = partielle(poly,K,l)
    plt.plot(x,y,'--',label='somme partielle, $L=$'+str(l))
plt.legend()
plt.title('Polynome trigo et sommes partielles de Fourier')
plt.subplot(122)
plt.plot(x, cren(x),label='fonction')
plt.ylabel('Amplitude')
for l in range(0,15,4):
    y = partielle(cren,K,l)
    plt.plot(x,y,'--',label='somme partielle, $L=$'+str(l))
plt.legend()
plt.title('Créneau et sommes partielles de Fourier')
plt.show()
```

```
/home/rtexier/anaconda3/lib/python3.9/site-packages/matplotlib/cbook/__init__.py:
1298: ComplexWarning: Casting complex values to real discards the imaginary part
  return np.asarray(x, float)
```

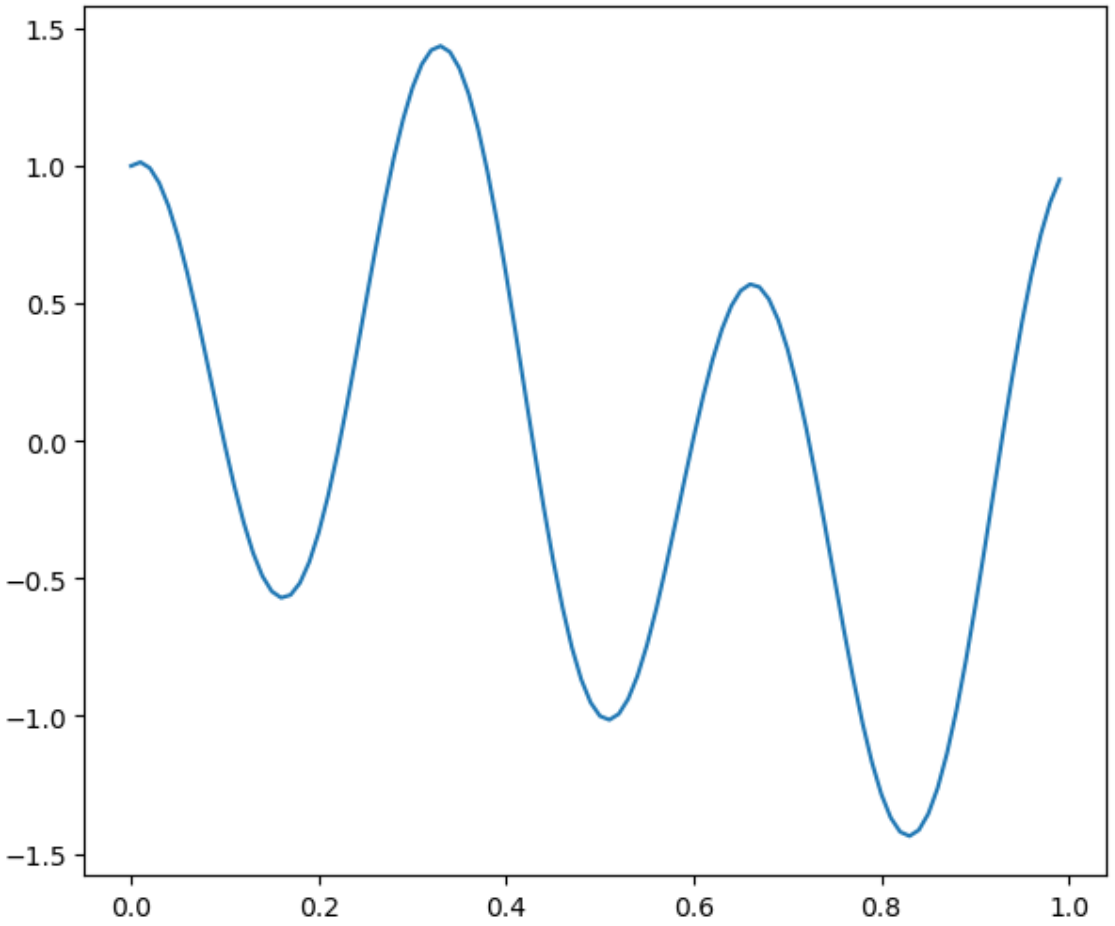


Partie 2 : Équation de la chaleur

Équation sans terme source ($g = 0$)

In [7]:
```
def partielle2(F,L):
# somme partielle de Fourier d'une fonction, connaissant les coefficients de Four
# L = nb de termes dans la somme partielle
    K = len(F)
    k = int(K/2)
    y = np.ones(2*k, dtype=complex)*F[k]
    x = np.linspace(0,1,K+1)[0:K]
    for l in range(1,L):
        y += F[k+l]*expo(x,l)+F[k-l]*expo(x,-l)
    return(y)

K = 100
k = int(K/2)
F = fourier(poly,k)
y = partielle2(F,k)
x = np.linspace(0,1,K+1)[0:K]
plt.figure(figsize = (7, 6))
plt.plot(x,y)
plt.show()
```

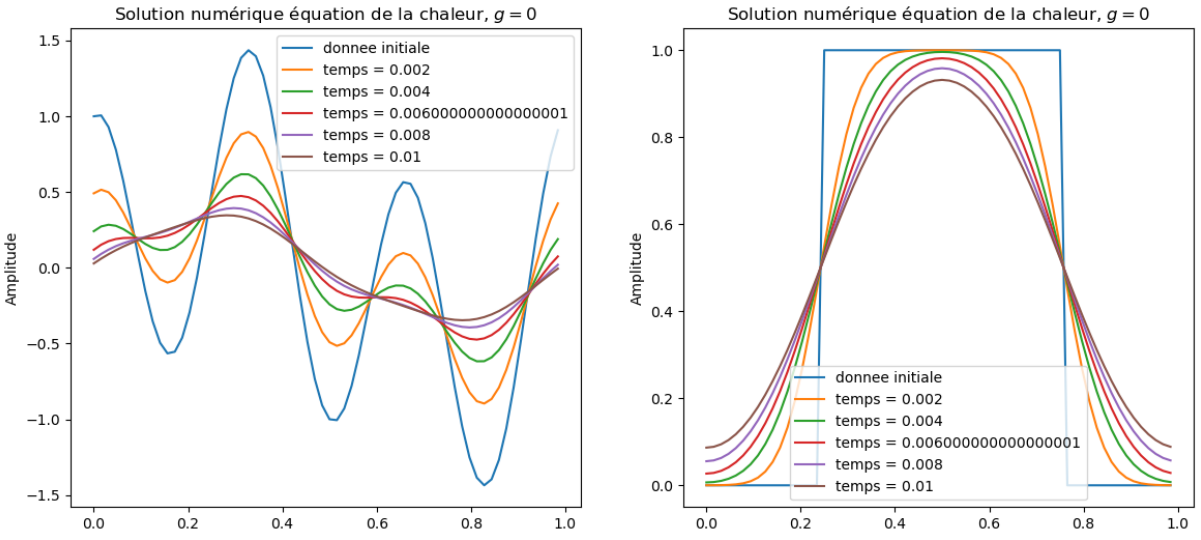


In [9]:
```
# Parametres
K = 64  # nombre de modes de Fourier
k = floor(K/2)
T = 0.01 # taille de l'intervalle de temps
p = 5 # T/p pas de la discrétisation en temps
t = np.linspace(0,1,p+1)*T

# Calcul de la somme partielle de Fourier de u
def sol_chaleur0(u0,K,T,L,p):
    # renvoie un tableau dont chaque ligne représente une discrétisation de la so
    # de la solution numérique, à un instant t donné
    F0 = fourier(u0,k).reshape((1,K))
    n = np.arange(-k,k).reshape((1,2*k))
```

```
    dt = T/p
    t = dt*np.arange(0,p+1).reshape((p+1,1))
    E = np.exp(-4*pi**2*t.dot(n**2))
    F = E * np.ones((p+1,1)).dot(F0)
    Y = np.zeros((p+1,2*k),dtype=complex)
    for i in range(p+1):
        Y[i,:] = partielle2(F[i,:],k)
    return(Y)

x = np.linspace(0,1,K+1)[0:K]
plt.figure(figsize = (14, 6))
plt.subplot(121)
plt.plot(x, poly(x),label='donnee initiale')
plt.ylabel('Amplitude')
Y = sol_chaleur0(poly,K,T,k,p)
for j in range(1,p+1):
    plt.plot(x,Y[j,:],label='temps = '+str(t[j]))
plt.legend()
plt.title('Solution numérique équation de la chaleur, $g=0$')
plt.subplot(122)
plt.plot(x, cren(x),label='donnee initiale')
plt.ylabel('Amplitude')
Y = sol_chaleur0(cren,K,T,k,p)
for j in range(1,p+1):
    plt.plot(x,Y[j,:],label='temps = '+str(t[j]))
plt.title('Solution numérique équation de la chaleur, $g=0$')
plt.legend()
plt.show()
```



Les graphes permettent d'observer deux phénomènes. D'abord, une tendance à l'homogénéisation au cours du temps : si u désigne une température, cela est cohérent avec le phénomène de diffusion de la chaleur des zones chaudes vers les zones froides. Ce phénomène conduit à réduire les écarts de température dans le milieu. Ensuite, on observe que pour tout $t > 0$, la solution $u(t, .)$ est régulière, ce qu'on peut aussi montrer par la théorie (effet régularisant du noyau de la chaleur). Du point de vue physique, la diffusion de chaleur conduit à lisser les sauts de température.

Évolution de l'énergie

On note

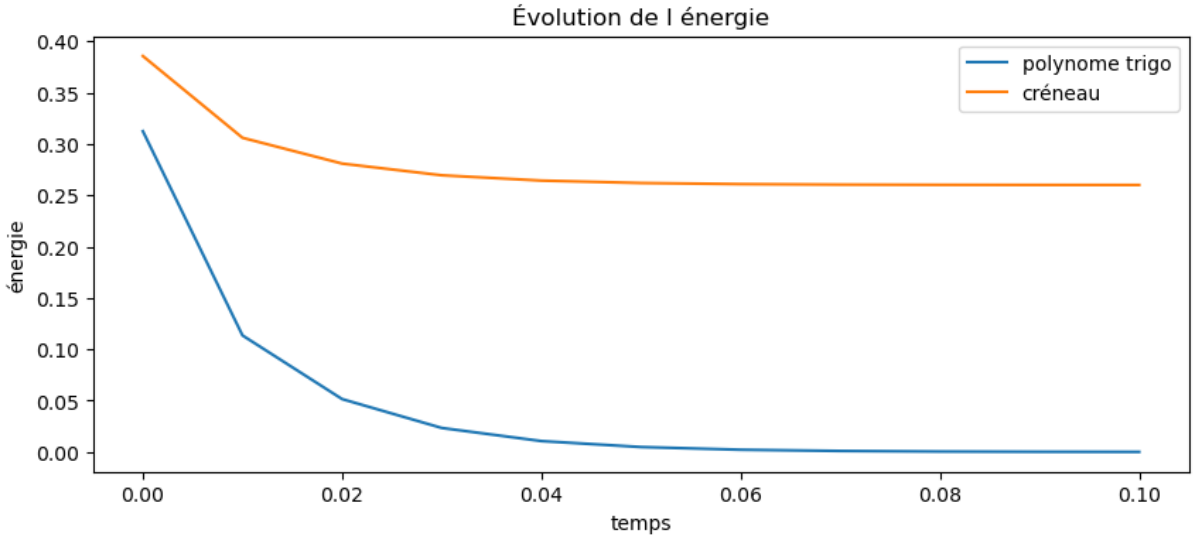
$$\mathcal{E}(t) = \sum_{-\frac{K}{2} \leq n \leq \frac{K}{2} - 1} |\hat{u}_n(t)|^2.$$

On rappelle que, du fait des propriétés de la transformée de Fourier discrète, cette énergie est égale à la norme L^2 de la fonction $u(t,\cdot)$. Si u désigne une température, cela fournit l'énergie thermique associée.

Calculons cette énergie pour nos deux données initiales.

In [10]:
```
def energie(u0,K,T,p):
    # renvoie un vecteur contenant l'énergie calculée à des instants t entre 0 et
    k = int(K/2)
    F0 = fourier(u0,k).reshape((1,K))
    n = np.arange(-k,k).reshape((1,2*k))
    dt = T/p
    t = dt*np.arange(0,p+1).reshape((p+1,1))
    E = np.exp(-4*pi**2*t.dot(n**2))
    F = E * np.ones((p+1,1)).dot(F0)
    Fn = np.abs(F)**2
    En = Fn.dot(np.ones(2*k))
    return(En)

T = 0.1
p = 10
E1 = energie(poly,100,T,p)
t = np.linspace(0,T,p+1)
plt.plot(t,E1,label='polynome trigo')
E2 = energie(cren,100,T,p)
plt.plot(t,E2,label='créneau')
plt.legend()
plt.xlabel('temps')
plt.ylabel('énergie')
plt.title('Évolution de l énergie')
plt.show()
```



On observe que l'énergie décroît dans les deux cas. Pour le polynôme trigo, qui est de moyenne nulle, l'énergie tend vers 0, tandis que pour le créneau, elle tend vers 0.25 qui est l'énergie de la fonction constante égale à 0.5.

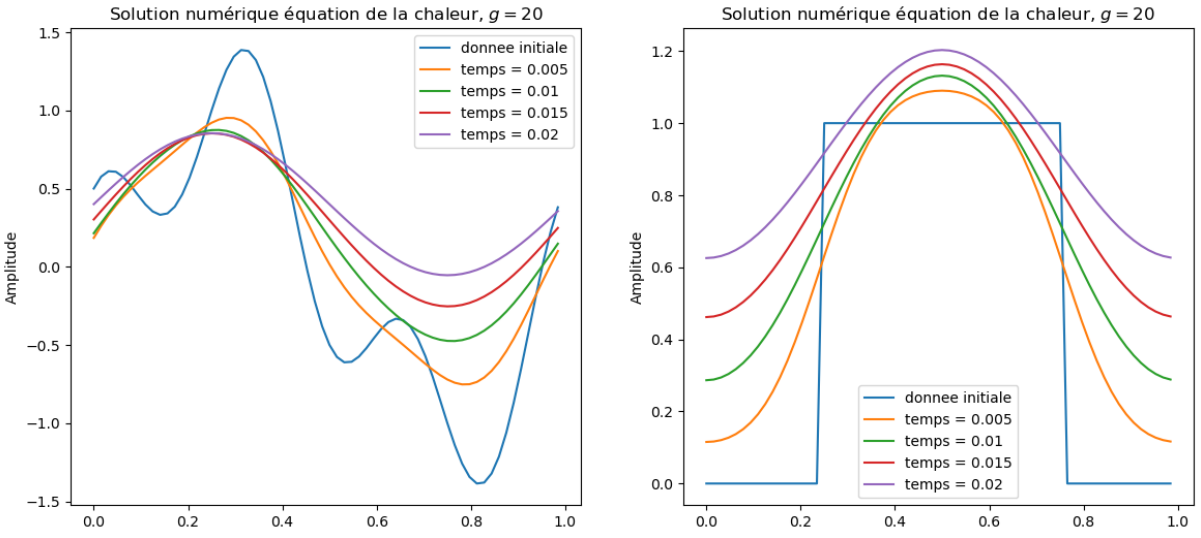
Équation avec terme source

In [13]:
```
# Parametres
K = 64  # nombre de modes de Fourier
k = floor(K/2)
T = 0.05 # taille de l'intervalle de temps
p = 10 # T/p pas de la discrétisation en temps
t = np.linspace(0,T,p+1) # discrétisation de l'intervalle de temps
```

```python
# Calcul de la somme partielle de Fourier de u
def sol_chaleur(u0,g,K,T,L,p):
    # renvoie un tableau dont chaque ligne représente une discrétisation de la so
    # de la solution numérique, à un instant t donné
    k = int(K/2)
    dt = T/p
    F0 = fourier(u0,k).reshape((1,K))
    G = np.zeros((p+1,K),dtype=complex)
    n = np.arange(-k,k).reshape((1,2*k))
    t = dt*np.arange(0,p+1).reshape((p+1,1))
    E = np.exp(-4*pi**2*t.dot(n**2))
    F = E * np.ones((p+1,1)).dot(F0)
    Y = np.zeros((p+1,2*k),dtype=complex)
    for i in range(p+1):
        def gt(x):
            return g(i*dt,x)
        G[i,:] = fourier(gt,k)
        Gi = E[1:i+1,:]*G[i:0:-1,:]
        F[i,:] += dt*(np.ones((1,i)).dot(Gi)).reshape(K,)
        Y[i,:] = partielle2(F[i,:],k)
    return(Y)

def cons(t,x):
    return 20*np.ones_like(x)
def nul(t,x):
    return np.zeros_like(x)

x = np.linspace(0,1,K+1)[0:K]
plt.figure(figsize = (14, 6))
plt.subplot(121)
plt.plot(x, poly(x),label='donnee initiale')
plt.ylabel('Amplitude')
Y = sol_chaleur(poly,cons,K,T,k,p)
for j in range(1,5):
    plt.plot(x,Y[j,:],label='temps = '+str(t[j]))
plt.legend()
plt.title('Solution numérique équation de la chaleur, $g=20$')
plt.subplot(122)
plt.plot(x, cren(x),label='donnee initiale')
plt.ylabel('Amplitude')
Y = sol_chaleur(cren,cons,K,T,k,p)
for j in range(1,5):
    plt.plot(x,Y[j,:],label='temps = '+str(t[j]))
plt.title('Solution numérique équation de la chaleur, $g=20$')
plt.legend()
plt.show()
```

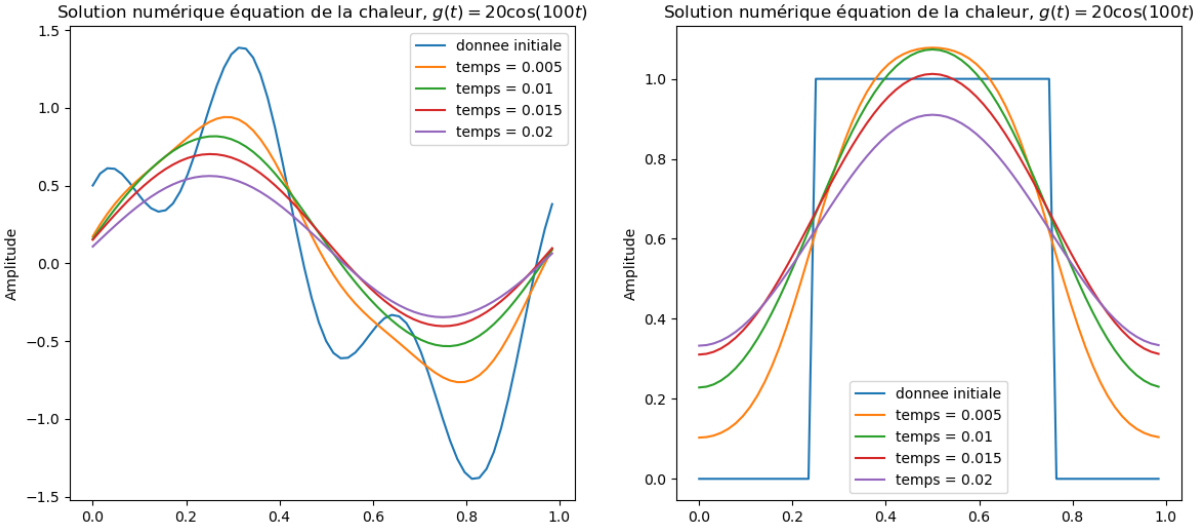


En plus de l'effet régularisant et de la tendance à l'homogénéisation observées précédemment, on peut remarquer ici que la valeur moyenne de u augmente régulièrement au cours du temps. Cela

découle du terme source (apport en chaleur).

In [14]:
```
x = np.linspace(0,1,K+1)[0:K]

def g2(t,x):
    return 20*np.cos(100*t)*np.ones_like(x)
plt.figure(figsize = (14, 6))
plt.subplot(121)
plt.plot(x, poly(x),label='donnee initiale')
plt.ylabel('Amplitude')
Y = sol_chaleur(poly,g2,K,T,k,p)
for j in range(1,5):
    plt.plot(x,Y[j,:],label='temps = '+str(t[j]))
plt.legend()
plt.title('Solution numérique équation de la chaleur, $g(t)=20 \cos(100t)$')
plt.subplot(122)
plt.plot(x, cren(x),label='donnee initiale')
plt.ylabel('Amplitude')
Y = sol_chaleur(cren,g2,K,T,k,p)
for j in range(1,5):
    plt.plot(x,Y[j,:],label='temps = '+str(t[j]))
plt.title('Solution numérique équation de la chaleur, $g(t)=20 \cos(100t)$')
plt.legend()
plt.show()
```



Ici la valeur moyenne de u augmente pour les temps compris entre 0 et $\frac{\pi}{200} \approx 0.015$, puis la tendance s'inverse car g change de signe.

In [ ]:
```
```