众所周知：大部分的偏微分方程很难找到解析解（符号解）。商业数学软件Matlab,Mathematica等均提供了功能强大的符号计算能力，但是对于数学上解不存在的PDE也是无能为力的。

1. 符号求解

首先，让我们以一维无线长细杆上的热传导问题为例

$$\begin{cases} u_t & = & a^2 u_{xx}, \\ u(x, t=0) & = & u_0(x). \end{cases}$$

由Fourier变换方法（假设）容易知道该问题的解析解为

$$u(x,t) = \int_{-\infty}^{+\infty} u_0(\xi) \left[\frac{1}{2a\sqrt{\pi t}} e^{-\frac{(x-\xi)^2}{4a^2 t}} \right] d\xi$$

若初始温度取为

$$u_0(x) = \begin{cases} 1 & (0 \le x \le 1) \\ 0 & (x < 0, x > 1) \end{cases}$$

注意到这就是一个在$[0,1]$上高度为1的矩形脉冲，故可简化解析解为

$$u(x,t) = \int_{0}^{1} \frac{1}{2a\sqrt{\pi t}} e^{-\frac{(x-\xi)^2}{4a^2 t}} d\xi$$

用如下序列观察该解析解：

In [1]:
```
a = 2
```

```
a =  2
```

In [103]:
```
x = -10:.5:10; t = 0.01:.1:1;  tau = 0:0.01:1;
[xx,tt,xi] = meshgrid(x, t, tau);
size(xx)
```

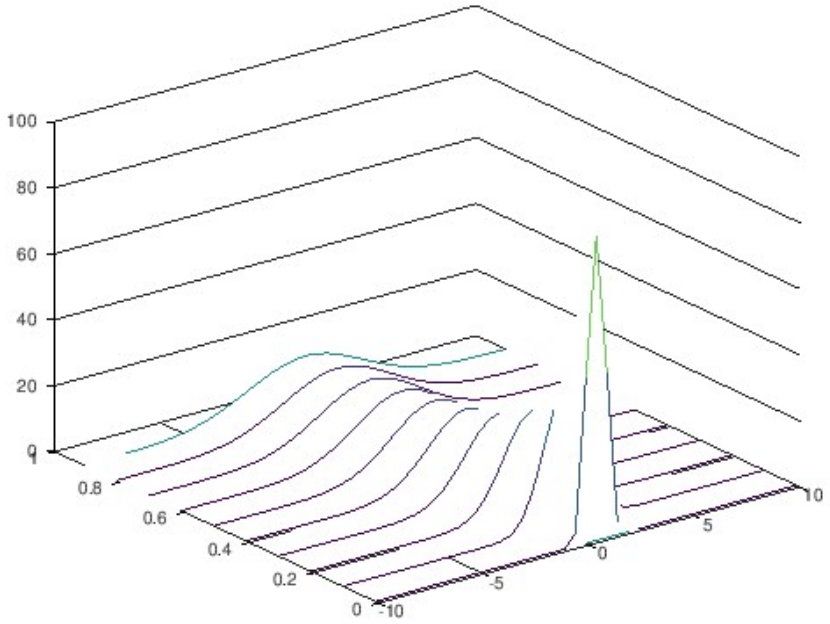
```
ans =

    10    41   101
```

In [102]:
```
fun = 1/2/a./sqrt(pi*tt).*exp(-(xx-xi).*(xx-xi)./(4*a*a*tt));
size(fun)
```

```
ans =

    10    41   101
```

In [128]:
```
uu=trapz(fun,3);     %% use quad()  instead, how to ?
```

In [129]:
```
waterfall(xx(:,:,1), tt(:,:,1), uu)
```



In [ ]:
```

```

我们在这里不可能也不敢过多深入寻求PDE分析解。事实上，即使寻求典型非线性方程的数值解也是困难的。

这一节中，我们旨在通过介绍简单PDE的最简单计算格式，重点在于展示octave的基于matrix概念的计算过程。这一过程有别于直接函数接口调用，也区别于C语言那样过多的编程细节

2.扩散过程数值模拟

物理扩散过程满足类似如下形式的“热传导/扩散”方程：

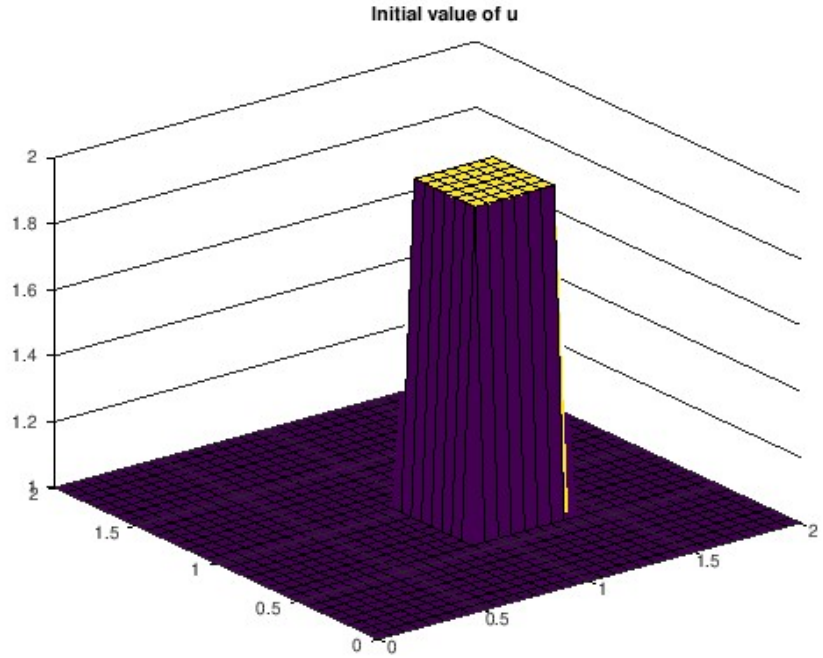
$$\frac{\partial u}{\partial t} = g(x,y)\frac{\partial^2 u}{\partial x^2} + g(x,y)\frac{\partial^2 u}{\partial y^2}, \quad \forall (x,y) \in \Omega$$

适当的边值条件如$\frac{\partial u}{\partial n} = 0$或$u = 0$在$(x,y) \in \partial\Omega$以及初始条件$u(0,x,y) = u_0(x,y)$必须给出以保证该方程可解。这里让我们略过各类“正则性条件”以及“相容性条件”等探讨。

In [1]:
```
% 考虑[0,2]x[0,2]上 g=0.05,  边界条件取1,
nx = 33; % 定义x方向节点数
ny = 33; % 定义y方向节点数
g = 0.05; % 扩散系数 - 这里取常数
dx = 2 / (nx - 1);
dy = 2 / (ny - 1);
dt = 0.25*dx*dy; t = 0;
```

In [2]:
```
% 初始条件任意挖空一个方块
x = linspace(0,2,nx); y = linspace(0,2,ny);
[xx,yy] = meshgrid(x,y);
```

In [6]:
```
u = ones(size(xx));    t = 0;    % 初值
u(floor(nx/3):ceil(nx/2), floor(ny/2):ceil(2*ny/3)) = 2;   % 初值突起
surf(xx,yy,u); title('Initial value of u');
```



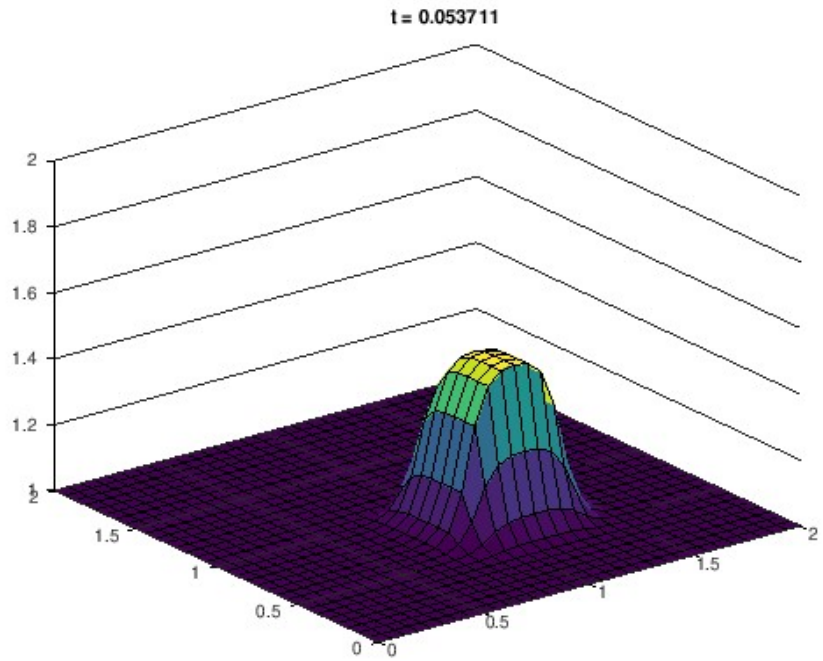
对于简单的情形$g(x,y)=\nu$为热扩散系数，在图像去噪中，$g(x,y)$是非常数，即为经典的P-M去噪模型。读者可以自行找到更多其它基于PDE的去噪模型，这里我们不展开讨论。对于图像去噪问题来说或，边界条件和计算格式的精度并不是本质重要的，这里我们采用了典型的“显格式”：

$$\frac{u_{i,j}^{n+1}-u_{i,j}^{n}}{\triangle t}=g_{i,j}\frac{u_{i+1,j}^{n}-2u_{i,j}^{n}+u_{i-1,j}^{n}}{\triangle x^2}+g_{i,j}\frac{u_{i,j+1}^{n}-2u_{i,j}^{n}+u_{i,j-1}^{n}}{\triangle y^2}$$

这样，化简之后可以得到递推式/迭代格式：

$$u_{i,j}^{n+1}=u_{i,j}^{n}+\frac{\triangle t}{\triangle x^2}g_{i,j}\left(u_{i+1,j}^{n}-2u_{i,j}^{n}+u_{i-1,j}^{n}\right)+\frac{\triangle t}{\triangle y^2}g_{i,j}\left(u_{i,j+1}^{n}-2u_{i,j}^{n}+u_{i,j-1}^{n}\right)$$

In [7]:
```
# 开始迭代/递推
nstep = 55;
for k = 1:nstep  % 迭代100步
    t = t + dt;

    taux = dt*g/dx/dx; tauy = dt*g/dy/dy;  % DO IT HERE if nonlinear problem is con
siered!

    un = u(:,[2:ny, 1]);        % 中心差法计算微分的"模板"
    us = u(:,[ny, 1:ny-1]);       %      WN  N  EN
    ue = u([2:nx, 1],:);          %      W   O  E
    uw = u([nx, 1:nx-1]);         %      WS  S  ES

    uxx = ue - 2*u + uw;         uyy = un - 2*u + us;

    u = u + taux*uxx + tauy*uyy;

end
        surf(xx,yy,u); title(['t = ', num2str(t)]); axis([0 2 0 2 1 2]);% pause(0.
5);
```



Python语言版本，可以参考：https://blog.csdn.net/qq_42000453/article/details/83097516 (https://blog.csdn.net/qq_42000453/article/details/83097516)

例：图像去噪的PDE模型探索

在上述抛物方程中，令u为图像的灰度值，为了与文献中的记号一致，在做图像处理过程中，我们会记未知函数为I来代替方程中的u。图像去噪问题中的经典Perona-Malik(P-M)模型

$$I_t = \nabla \cdot (g(|DI|)\nabla I)$$

扩散系数/函数取为 $g(\cdot)=\frac{1}{1+(\frac{x}{K})^2}$ 或 $g(\cdot)=e^{-(\frac{x}{K})^2}$。此处常数$K$是一个阈值（Edge Stop参数），与图像的品质有关。下面是一段自动计算K数值的程序，有兴趣的读者可以搜索相关文献得到更丰富类型的模型推广。

In [8]:
```
function K = autoK(I)
% 自动估计梯度阈值K函数
% I:  input gray or color image
[row,col,nchannel] = size(I);  K = 0;
if nchannel == 1     %gray image
    [gradx,grady] = gradient(I);
    gradI = (gradx.^2 + grady.^2).^0.5;
    % 用robust_statistic自动估计梯度阈值(参Sapiro P231)
    K = 1.4826*mean(mean(abs(gradI-mean(mean(gradI)))));
else%color image
    for i=1:3
        [gradx,grady] = gradient(I(:,:,i));
        gradI = (gradx.^2 + grady.^2).^0.5;
        K = K + 1.4826*mean(mean(abs(gradI - mean(mean(gradI)))));
    end
    K = K/3;
end
endfunction
```

首先测试autoK函数：

In [9]:
```
I = imread('img/lena256.bmp');imshow(I)
K = autoK(I)
```

```
K =  3.1212
```

In [11]:
```
I = double(I)/256; size(I)
```

```
ans =

   256   256
```

In [12]:
```
I(1:8,1:8)
```

```
ans =

 Columns 1 through 6:

   0.0020752   0.0020752   0.0020905   0.0020905   0.0021210   0.0020447
   0.0020752   0.0020752   0.0020905   0.0020905   0.0021210   0.0020447
   0.0020752   0.0020752   0.0020905   0.0020905   0.0021210   0.0020447
   0.0020294   0.0021362   0.0020294   0.0019989   0.0020294   0.0019531
   0.0019531   0.0020142   0.0020294   0.0019684   0.0021057   0.0020142
   0.0019836   0.0019989   0.0019836   0.0019226   0.0020142   0.0019836
   0.0019836   0.0019989   0.0019836   0.0020142   0.0020599   0.0020142
   0.0019989   0.0020294   0.0019531   0.0020294   0.0019379   0.0018616

 Columns 7 and 8:

   0.0019684   0.0020447
   0.0019684   0.0020447
   0.0019684   0.0020447
   0.0019531   0.0019836
   0.0020599   0.0019531
   0.0020294   0.0019989
   0.0019531   0.0019531
   0.0019531   0.0019531
```

In [13]:
```
% 添加噪声产生测试图像:
H = [0.011344 0.083820 0.011344; ...
     0.083820 0.619347 0.083820; ...
     0.011344 0.083820 0.011344];
%% alternative way:
%pkg load image
%H = fspecial('gaussian',3,0.5)
```

In [14]:
```
Ig = I;
```

In [15]:
```
Ig = conv2(Ig, H, 'same');    %%%  执行二维卷积，等价于添加Gauss噪声或光滑化
```

关于图像的卷积运算，我们引用来自于网络的图片加以解释：
conv1
卷积后图像的尺寸会有所不同，函数conv2中提供的参数'same'可保持图像尺寸，当然也可以有其他选项，我们在讲卷积神经网络的时候将会介绍更多

conv2

In [17]:
```
Ig = conv2(Ig, H, 'same');    %%%  可执行若干次二维卷积，I-Ig将更加明显，这里只用与光滑化
```

In [18]:
```
imshow([I Ig I - Ig])    % 此时I, Ig均为浮点数，应由int8转换为0-255灰度后显示效果更真实
```

肉眼很难区分，可以用信噪比这个数据来量化图像的清晰程度。信噪比，英文名称叫做SNR或S/N（(Signal-Noise Ratio)），又称为讯噪比。顾名思义，它是度量信号的平均功率和噪声的平均功率之比的一种方式，物理上被定义为10倍对数信号与噪声功率比。对于分辨率为$m \times n$的数字图像I来说

$$SNR(I, I_n) = 10 \log_{10} \left[\frac{\sum_{i=1}^{m} \sum_{j=1}^{n} I(i,j)^2}{\sum_{i=1}^{m} \sum_{j=1}^{n} [I(i,j) - I_n(i,j)]^2} \right],$$

其中，I_n是含噪声图像。它的单位是分贝。下面是计算信噪比的一个实现：

In [19]:
```
function snr = SNR(I, In)
% 计算信号噪声比函数，其中
% I : 原图像
% In: 带噪声图像

info = I - mean(mean(I));    % 原图本身方差水平

dI = I - In; noise = dI - mean(mean(dI)); % 噪声方差水平

% 信噪比:
snr = 10*log10(sum(sum(info.*info))/sum(sum(noise.*noise)));

%% 为节省篇幅，处理彩色图像请自行补全（参考autoK的做法）
endfunction
```

In [20]:
```
SNR(I,Ig)
SNR(double(I),double(Ig))
```

```
ans =  15.005
ans =  15.005
```

In [179]:
```
%  换用其他下载的(原图与噪声配套)图像单独测试SNR函数
I = imread('img/I_lenna.jpg');
Ig = imread('img/I_lenna_noise.jpg');
imshow([I Ig I-Ig])
```

In [180]:
```
I = double(I); Ig = double(Ig);    % this is important
```

In [181]:
```
In = Ig;  % save the noise image for comparision later
```

In [186]:
```
Ig = conv2(Ig, H, 'same');
```

In [187]:
```
snr = SNR(I, Ig)
```

```
snr =  12.490
```

此外，可用于度量数字图像质量的指标还有：峰值信噪比(Peak Signal to Noise Ratio,PSNR)和结构相似性(Structural SIMilarity, SSIM)等，在图像增强和图像分割效果度量中有重要作用。这里我们不一一赘述。

In [59]:
```

```

下面，让我们开始执行计算来取出Gauss噪声,并观察SNR在此过程中的变化!

In [210]:
```
for tt = 1:100    % do this repeatedly until satisfied!
 %% 1. 计算(N,S,E,W)四方向的梯度
    %    N
    % W  O  E
    %    S
    Gn = [Ig(1,:,:);Ig(1:end-1,:,:)] - Ig;   % N - O
    Gs = [Ig(2:end,:,:);Ig(end,:,:)] - Ig;   % S - O
    Ge = [Ig(:,2:end,:) Ig(:,end,:)] - Ig;   % E - O
    Gw = [Ig(:,1,:) Ig(:,1:end-1,:)] - Ig;   % W - O

 %% 2. 计算扩散稀疏
     Cn = 1./(1 + (Gn/K).^2);
     Cs = 1./(1 + (Gs/K).^2);
     Ce = 1./(1 + (Ge/K).^2);
     Cw = 1./(1 + (Gw/K).^2);

%% 3. 一阶时间格式：求解PDE(扩散)
    diff = (Cn.*Gn + Cs.*Gs + Ce.*Ge + Cw.*Gw);
    dt = 0.05;    %% 可视为某种超参数， 在PDE数值解中可理解为"时间步长"
    Ig = Ig + dt*diff;    % 这是一个显欧拉格式


    %% show the image or calculate SNR
    % imshow([In Ig I]); SNR(I, Ig)
 end
```

In [211]:
```
imshow([In Ig I]); SNR(I, Ig)
```

```
ans =  12.568
```

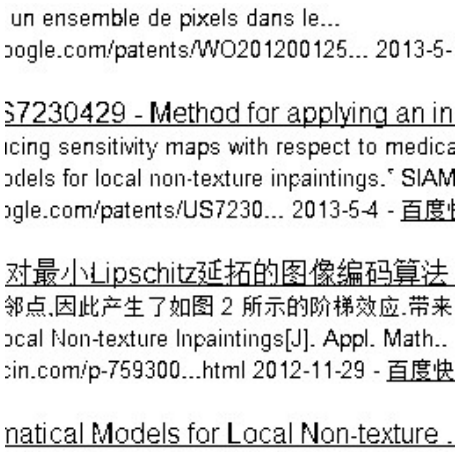
请注意：信噪比越高，说明有效原始信号比重越高，而噪声的比重就越低！

Remark & Suggestion

前面所展示的是一个最基本的P-M模型，实际更有效的去噪模型，需要对方程的非线性形式（主要是g的取法或增加对流项）、迭代的数值格式(这里是显Euler格式)、超参数(时间步长等)以及去噪过程中配合去模糊以保证SNR的前提下增强可视化效果。更多的细节，请读者翻阅数字图像处理专业期刊: IEEE Trans on Image Processing 1995-2015年间的论文。

例： 让我们来看一个基于变分原理的图像去水印例子，其中变分极小化过程在数学上等价于PDE求解

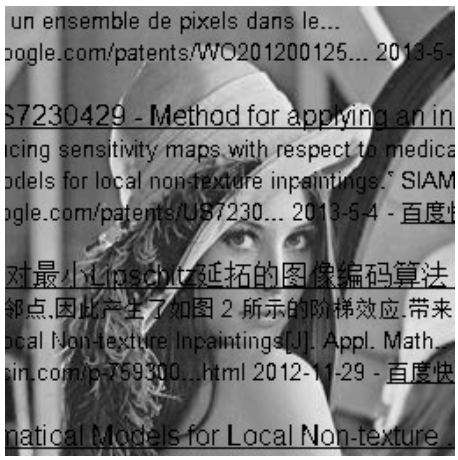
In [21]:
```
clear;
img = double(imread('lena.jpg'));
mask= rgb2gray(imread('ma.jpg'))>160;
imshow(mask)
```



值得指出的是，我们需要一个mask，即“遮罩图像”，以方便程序确定哪些地方需要被修复。这是一种简便的做法，更通用的设计将会使程序变得异常复杂。

这里，我们用如上这张文字图像(为方便起见，尺寸与lena图一致)。在这个例子中，我们现场合成需要修复的图像，原则上读者可以用任意方法加入“水印图像”：

In [22]:
```
[m n]=size(img);
for i=1:m
    for j=1:n
        if mask(i,j)==0
           img(i,j)=0;
        end
    end
end
imshow(uint8(img));    %% 此时img仍然保持浮点数类型，且被加了水印
```



接下来，考虑去掉文字水印

In [23]:
```
lambda = 0.2;
a = 0.5;
imgn = img;
iter = 0;
```

In [26]:
```
for k = 1:10  %迭代修复10次
    iter = iter + 1;
    fprintf('Round %d ... \n',iter);

    for i = 2:m-1
        for j = 2:n-1

            if mask(i,j)==0      %如果当前像素是被污染的像素，则进行处理

                Un=sqrt((img(i,j)-img(i-1,j))^2 + ((img(i-1,j-1)-img(i-1,j+1))/2)^
2);
                Ue=sqrt((img(i,j)-img(i,j+1))^2 + ((img(i-1,j+1)-img(i+1,j+1))/2)^
2);
                Uw=sqrt((img(i,j)-img(i,j-1))^2 + ((img(i-1,j-1)-img(i+1,j-1))/2)^
2);
                Us=sqrt((img(i,j)-img(i+1,j))^2 + ((img(i+1,j-1)-img(i+1,j+1))/2)^
2);


                Wn=1/sqrt(Un^2+a^2);
                We=1/sqrt(Ue^2+a^2);
                Ww=1/sqrt(Uw^2+a^2);
                Ws=1/sqrt(Us^2+a^2);

                Hon=Wn/((Wn+We+Ww+Ws) + lambda);
                Hoe=We/((Wn+We+Ww+Ws) + lambda);
                How=Ww/((Wn+We+Ww+Ws) + lambda);
                Hos=Ws/((Wn+We+Ww+Ws) + lambda);

                 %% 本质上，这里与去噪的P-M模型方程一致!
                Hoo = lambda/((Wn+We+Ww+Ws) + lambda);
                imgn(i,j) = Hon*img(i-1,j) + Hoe*img(i,j+1) + How*img(i,j-1) + Hos*
img(i+1,j) + Hoo*img(i,j);

            end
        end
    end
    %% 上述两重循环 + if判断，可以用向量化编程方式完成，请改写!
    img = imgn;
end
```

```
Round 11 ...
Round 12 ...
Round 13 ...
Round 14 ...
Round 15 ...
Round 16 ...
Round 17 ...
Round 18 ...
Round 19 ...
Round 20 ...
```

展示修复效果:

In [25]:
```
imshow(uint8(img));
```

Too slow, please run it in octave or matlab GUI. original image and inpainted result after 100 iteration:

In [15]:
```
function x = snr(ref, sig)
    mse = mean((ref(:)-sig(:)).^2);
    dv = var(ref(:),1);
    x = 10*log10(dv/mse);
endfunction
```

In [19]:
```
ref = double(imread('lena.jpg'));
snr(ref, img)
```

```
ans =  2.5267
```

练习1： 如程序中注释，请将上述循环中关于图像像素的行列循环改写成向量化形式，并列出两个版本的时间开销对比。

练习2：请去除下图中的文字水印：

In [14]:

```
ans =

   128   128
```

3. 流体的数值模拟初步

仅作介绍，不建议深入研究，参考simulation_fluid.ipynb。

In [ ]: