Lecture 30
JD-MCMC for Object Detection
ECEN 5283 Computer Vision

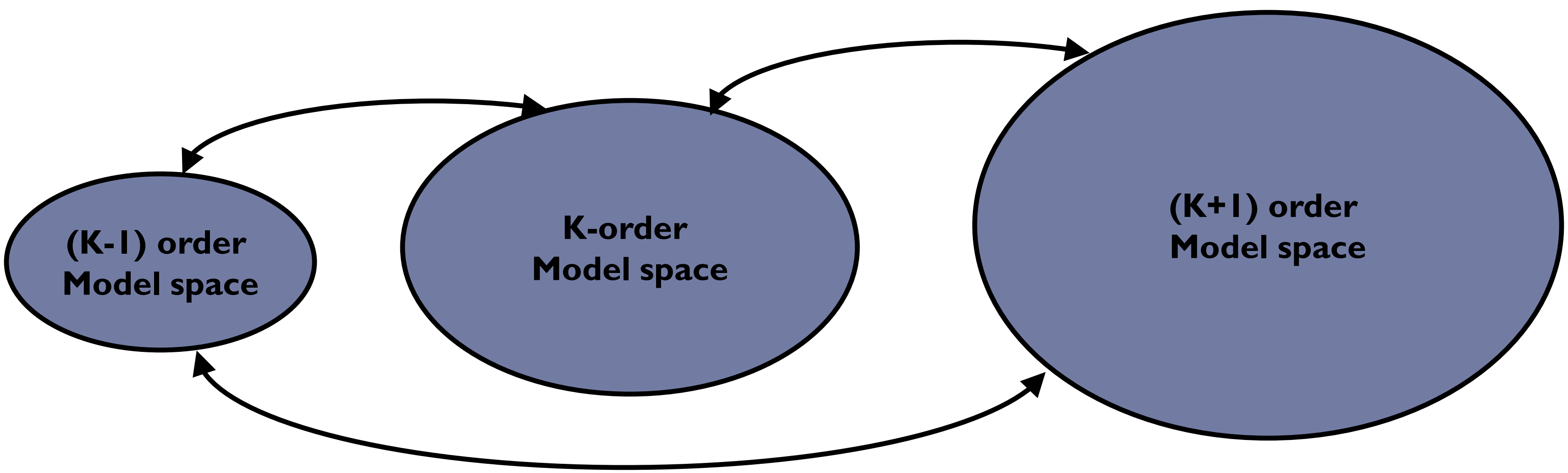
Dr. Guoliang Fan
School of Electrical and Computer Engineering
Oklahoma State University

Goal

- To apply JD-MCMC for object detection.
- To get ready for Project 5 by showing some Matlab examples.

Jump-Diffusion MCMC

- Jump-diffusion provide a mixed mechanism to draw samples from a disconnected state space where both discrete and continuous state variables exist.
  - Jump contributes in sampling over the parameter number.
    - Can be controlled by a probability
  - Diffusion contributes in sampling over the parameter values.
    - Can be managed by a random walk.



The new objective function

- Given a prior probability of model order *k* and an observed image Y, the solution of object detection (i.e., Θ: object locations) is represented by the joint posterior probability density as:

$$p(\Theta, k \mid Y) \propto \underline{p(Y \mid \Theta)}\,\underline{p(k)}$$

Unknown posterior density of the object number and locations

data likelihood given model Θ

Prior for model order k

Application to Object Detection

- There are two kinds of parameters
  - The number of objects, *k*,
  - The location of each object

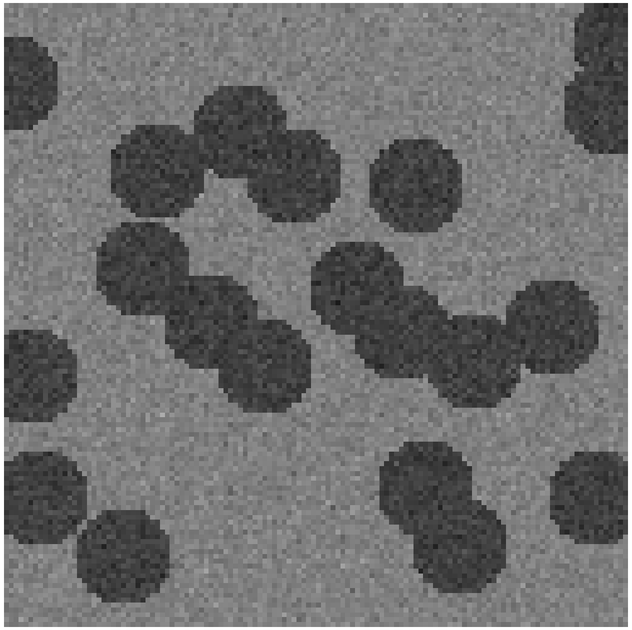
$$\Theta_k = \{(x_i, y_i) \mid i = 1, \ldots, k\}$$

- Two probabilistic functions
  - The prior probability of model order

$$p(K^* = k) = \frac{\lambda^k}{e^{\lambda} k!}$$

  - The likelihood of object detection given the order number

$$p(Y \mid \Theta) \propto \exp\left(- \frac{\left\| \mathbf{I} - \mathbf{J}(\Theta) \right\|}{2\sigma^2} \right)$$



Jump Diffusion MCMC Algorithm

- Initialize locations of *k* hypothesized objects and the maximum order K_{max}.
- for i=1:*N*
  - Draw a sample a~U(0,1)
  - If $a<0.33$ and $k>1$ (jump by -1)
    - k=k-1;
    - MCMC Gibbs sampling
    - Accept or reject by Metropolis Sampling
  - else if $a<0.66$ and $k<Kmax$ (jump by +1)
    - k=k+1;
    - MCMC Gibbs sampling
    - Accept or reject by Metropolis Sampling
  - else (no jump)
    - MCMC Gibbs sampling
    - Accept or reject by Metropolis Sampling
- End
- Select samples after M iterations (burn-in);
- Obtain a set of samples with certain step size.
- Compute the mean estimate of the object number k*
- For samples with k*, re-order all object locations and compute the mean location for each object.

$$p(\Theta, k \mid Y) \propto p(Y \mid \Theta)\, p(k)$$

PDF of interest used for evaluation

$$\alpha = \min\left(\frac{p(Y \mid \Theta^{(B)})P(k^{(B)})}{p(Y \mid \Theta^{(A)})P(k^{(A)})}, 1\right)$$

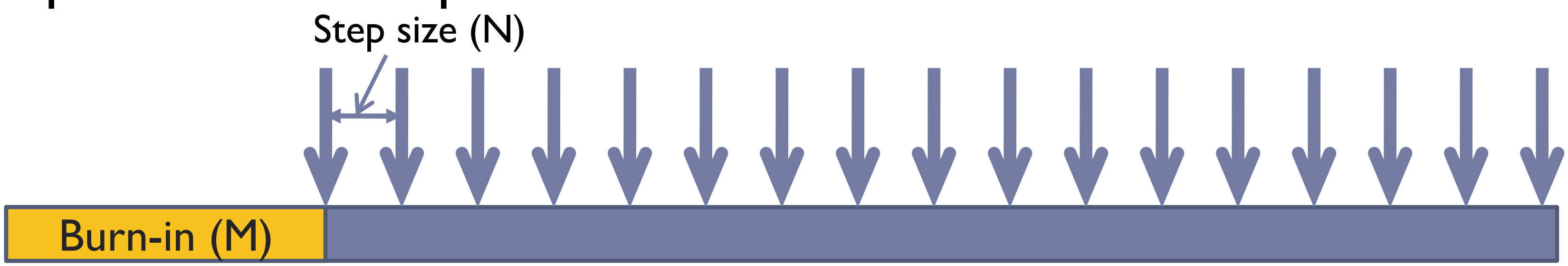
(acceptance probability for jump)

$$\beta = \left(\frac{p(Y \mid \Theta^{(B)})}{p(Y \mid \Theta^{(A)})}, 1\right)$$

(acceptance probability for no jump)

How to get the final solution?

- After enough sampling, we can use "burn-in" to throw away *M* samples in the beginning, and only use the later samples with step size *N* to compute the solution.



- Then we do a mean estimation for selected samples to find the unique deterministic solution.

$$k^* = \text{Round}\left(\frac{1}{L}\sum_{i=1}^{L} k_{(M+Ni)}\right) \quad (k_i : \text{the } i\text{th sample of the object number})$$

$$\bar{\mathbf{x}}^{(l)} = \text{Average}\left(\mathbf{x}^{(l)}_{k_i=k^*}\right) \quad (\mathbf{x}^{(l)}_{k_i=k^*} : \text{the } l\text{th object' position of the } i\text{th sample with } k^* \text{ order})$$

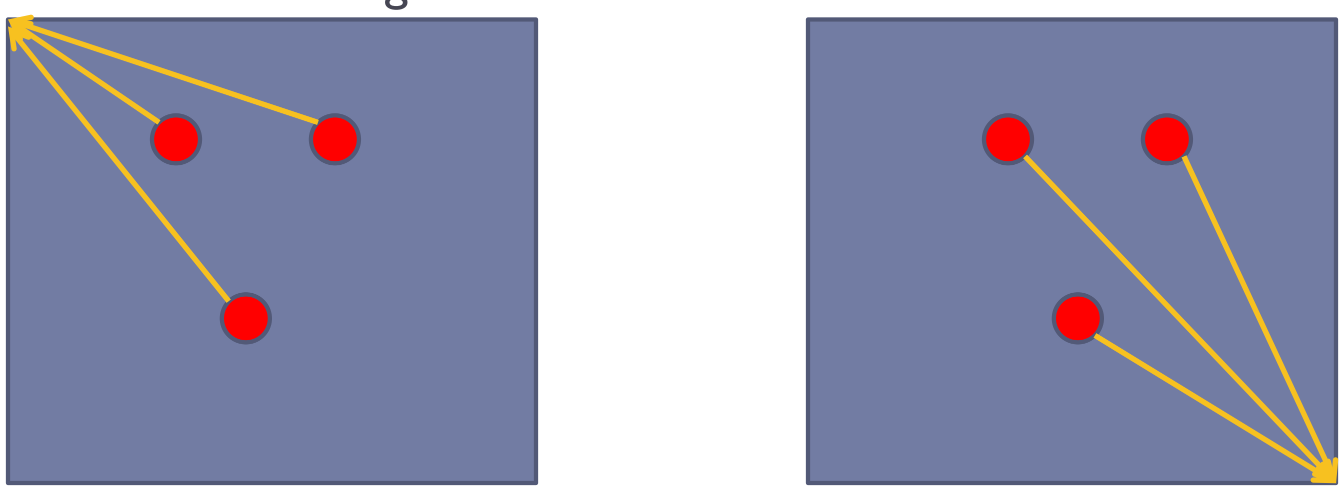
(Note: It is necessary to re-order all objects in the list of each sample to make sure the average is correct)

Object Location Re-ordering

- To make sure the mean estimation for object position is correct, a re-ordering is needed for the position list associated with each sample (with order k*) before computing the average.

$$\bar{\mathbf{x}}^{(l)} = \text{Average}\left(\mathbf{x}^{(l)}_{k_i=k^*}\right) \quad (\mathbf{x}^{(l)}_{k_i=k^*} : \text{the } l\text{th object' position of the } i\text{th sample with } k^* \text{ order})$$

- This can be done by setting certain rule of object ordering:
  - For example, all objects are re-ordered according to their distances to the top-left or bottom-right corner.



A few useful Matlab functions

(1) function L=likelihood(Image,Object,Locations,Num)

% This function computes the likelihood of current hypotheses about the number and locations.

% "Image" represents the gray-scale test image and "Object" is the object template.

% Locations is a 1*2Num vector saving the coordinates of Num objects.

(2) function drawcricle(Image,Locations,Num)

% This function draws Num circles in "Image" according to current location estimation.

(3) function N=clip(Locations,Mmin,Mmax)

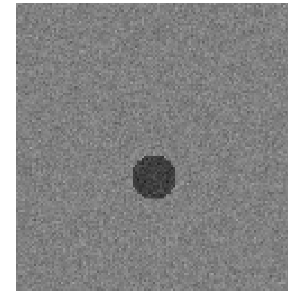
% This function will ensure the object locations are within the image.

% "Locations" save original coordinates, Mmin and Mmax are the minimum or maximum values.

(4) Matlab program "create.m" can be used to create a test image with certain number of objects.

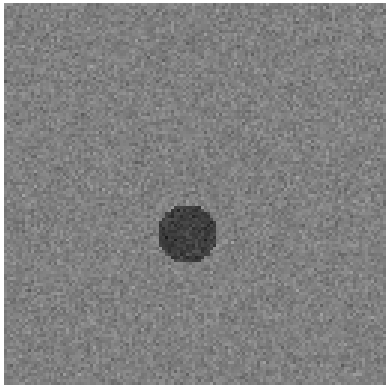
(5) An useful Matlab function is POISSPDF(k,lambda), P = POISSCDF(X,LAMBDA) computes the Poisson probability mass function with parameter LAMBDA at the values in X.

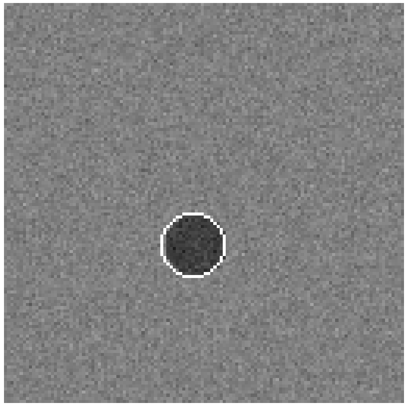
Matlab Code (1): Importance Sampling (importantsampling.m)

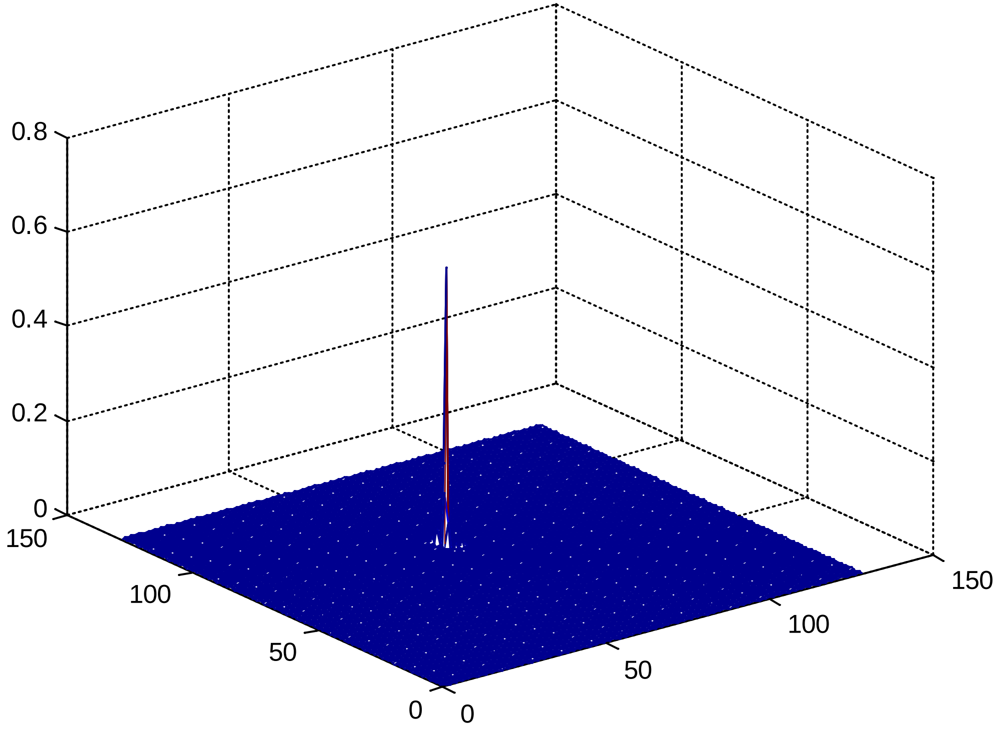


```
I=double(imread('discs1.bmp'))/255;          % read the test image
T=double(imread('target.bmp'))/255;          % read the target image
[X Y]=size(I);    Z=zeros(X,Y);              % the size of the image
Kmax=2000;                                   % the number of samples
Posi=[0 0];                                  % position variable

for i=1:Kmax
   Axy=round(rand(1,2)*X)+1;                 % draw a 2-D position hypothesis uniformly in the image
   Li(i)=likelihood(I,T,Axy,1);              % evaluate the likelihood
   Px(i,:)=Axy(1:2);                         % save the 2-D position hypothesis
   Z(Axy(1),Axy(2))=Li(i);                   % save the likelihood for that 2-D position hypothesis
   Posi=Posi+Li(i)*Px(i,:);                  % compute the weighted mean estimation
end

Posi=round(Posi/sum(Li));                    % compute the weighted mean estimation
Z=Z/sum(Li);                                 % compute the normalized distribution
J=drawcircle(I,Posi,1);                      % locate the object according to the mean estimation
figure(1), mesh(Z);                          % draw the estimated distribution
figure(2), imshow(J);                        % show the object detection result
```

Importance Sampling Examples



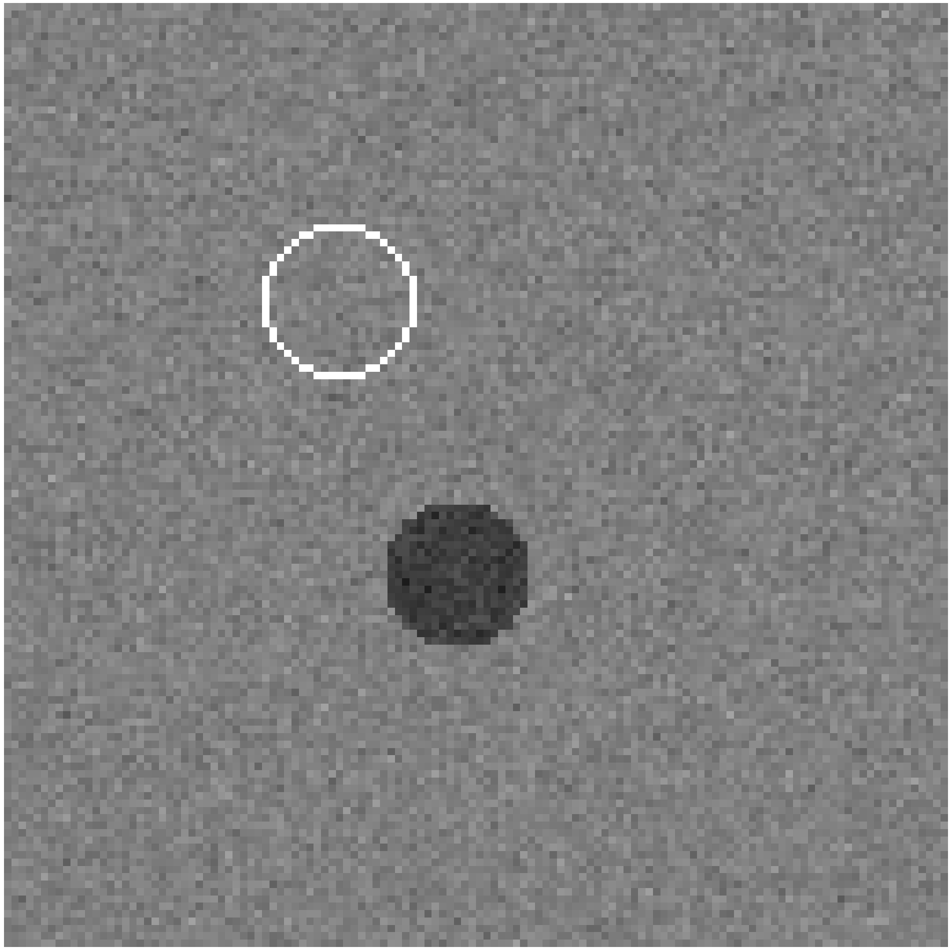




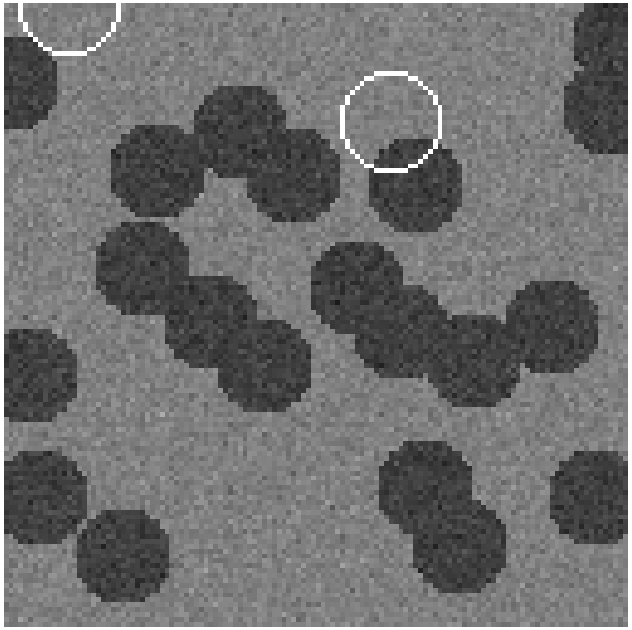
Matlab Code (2): Metropolis Sampling for MCMC (MCMC.m)

```
I=double(imread('discs1.bmp'))/255;                    % read the test image
T=double(imread('target.bmp'))/255;                    % read the target image
[X Y]=size(I); Z=zeros(X,Y);                           % the size of the input image
Kmax=100;                                              % the number of steps of random walks
Oxy=round(rand(1,2)*X)+1;                              % an initial position
L1=likelihood(I,T,Oxy,1);                              % the initial likelihood
Io=drawcircle(I,Oxy,1), ,imshow(Io);                   % locate the object in an image
Imframe(1:X,1:Y,1)=Io; Imframe(1:X,1:Y,2)=Io; Imframe(1:X,1:Y,3)=Io;
videoseg(1)=im2frame(Imframe);                         % make the first frame
for i=1:Kmax
   Dxy=Oxy+round(randn(1,2)*30);                       % random walk the standard deviation 30
   Dxy=clip(Dxy,1,X);                                  % make sure the position are within the image
   L2=likelihood(I,T,Dxy,1);                           % evaluate the likelihood
   v=min(1,L2/L1);                                     % compute the acceptance ratio
   u=rand;                                             % draw a sample uniformly in [0 1]
   if v>u
      Oxy=Dxy;    L1=L2;                               % accept the move
      Io=drawcircle(I,Oxy,1);                          % draw the new position
   end
   figure(1),imshow(Io);
   Imframe(1:X,1:Y,1)=Io; Imframe(1:X,1:Y,2)=Io; Imframe(1:X,1:Y,3)=Io;
   videoseg(i+1)=im2frame(Imframe);
end
movie2avi(videoseg(1:(Kmax+1)),'MCMC1.avi','FPS',1,'COMPRESSION','None');
```

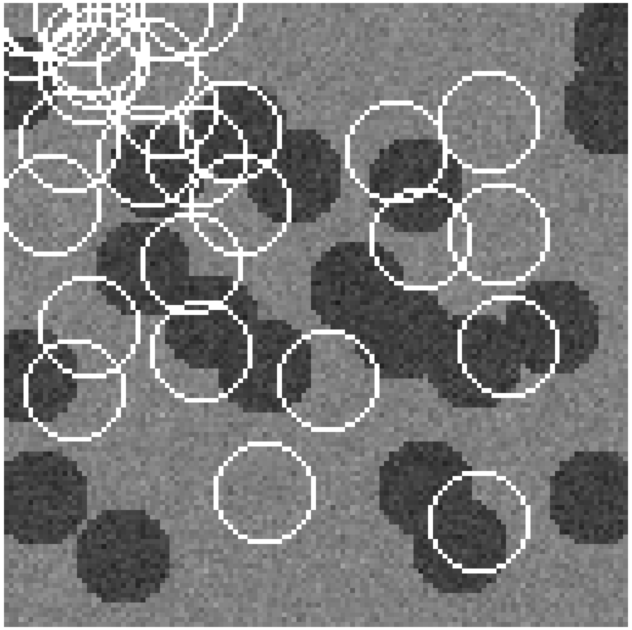
Metropolis Sampling Example



Object Detection Example



$k_0 = 2$



$k_0 = 30$