Today:

* Go through the algorithm for simulated annealing (SA)
  - Bonus! We get the Metropolis-Hastings for free! So we will go through this as well.
* Structure of first exam.
* Look at Exercises #4 This is a coding exercise (Pseudocodes based on the exercises are examinable).

Pseudocode / Algorithm for SA (§18.3):

18.3 Statement of Algorithm

Algorithm 7 Simulated Annealing Algorithm

Choose an initial guess $x^{(0)}$. Initialize $x = x^{(0)}$.
Select the temperature change counter $k = 0$
Select a cooling schedule T_k
Select an initial temperature $T = T_0 \geq 0$
Select a repetition schedule M_k, that defines the number of iterations executed at each temperature T_k
← At temp. T_k we generate M_k proposals
while Stopping criterion is not met **do**
← "While the cost function is still too large, keep going..."
  Set repetition counter $m = 0$
  while $m < M_k$ **do**
  ← generate M_k new proposals
    Generate a new state x'
    Calculate $\Delta E = E(x') - E(x)$.
    If $\Delta E \leq 0$, accept the new state, $x \leftarrow x'$ with probability 1.
    If $\Delta E > 0$, accept the new state, $x \leftarrow x'$, with probability $e^{-\Delta E/T_k}$.
    $m \leftarrow m + 1$
  end while
end while
← Missing step! Go from temperature T_k to T_{k+1}.

Inner loop

Inner loop is the Metropolis-Hastings (MH) algorithm, can be used to generate numbers from any continuous distribution $p(x)$.

Pseudocode for MH:

Pseudocode for MH:

- Initialize the state of the system as $\underline{x}_0$.
- Set an iteration counter $m = 1$
- Set sigma, set max iterations.
- while $m <$ max iterations
  - Generate a new proposal

    $\underline{x}_1 = \text{norm}(\underline{x}_0, \text{sigma})$
  - Compute the ratio of probabilities:

    $\text{accept\_prob} = p(\underline{x}_1) / p(\underline{x}_0)$
  - Generate

    $u = \text{uniform}(0,1)$
  - If $u <$ accept_prob
    - Accept new proposal
    - $\underline{x}_0 \leftarrow \underline{x}_1$

    Otherwise
    - Reject new proposal
    - $\underline{x}_0$ stays the same
- Store the value $\underline{x}_0$ in an array:

  $X[k] \leftarrow \underline{x}_0$

- End while loop.

Comment: Usually people run the MH algorithm for a number of steps (= n_burnin) before storing any values in the array X. Reason: so the values in X aren't influenced by the initial state. This works because the MH algorithm is a Markov-Chain process.

End of first part of Course! After today:

- ACM41030 — stay with me
- ACM40990

Structure of first exam: See list on website.

Exercises #4, Question 1 — Done in class, but see highlight below:

```
% ***************************************************
% MH routine

for i = 1:nsamp
    % Update chain:
    [x,a] = MHstep(x,sig);
    % Track accept-reject status
    acc = acc + [a 1];
    % Store the i-th sample:
    X(i) = x;
end

end

% ***************************************************

function [x1,a] = MHstep(x0,sig)
    % Generate candidate from Gaussian:
    xp = normrnd(x0,sig,1);
    % Compute acceptance probability:
    accprob = targetdist(xp) / targetdist(x0);
    u = rand; % uniform random number
    if u <= accprob % if accepted
```

$p(x) = \begin{cases} \lambda e^{-\lambda x} & x \geq 0 \\ 0 & x < 0 \end{cases}$

$u \in [0,1]$


```
            x1 = xp; % new point is the candidate
            a = 1; % note the acceptance
        else % if rejected
            x1 = x0; % new point is the same as the old one
            a = 0; % note the rejection
        end
    end

    % ...
```