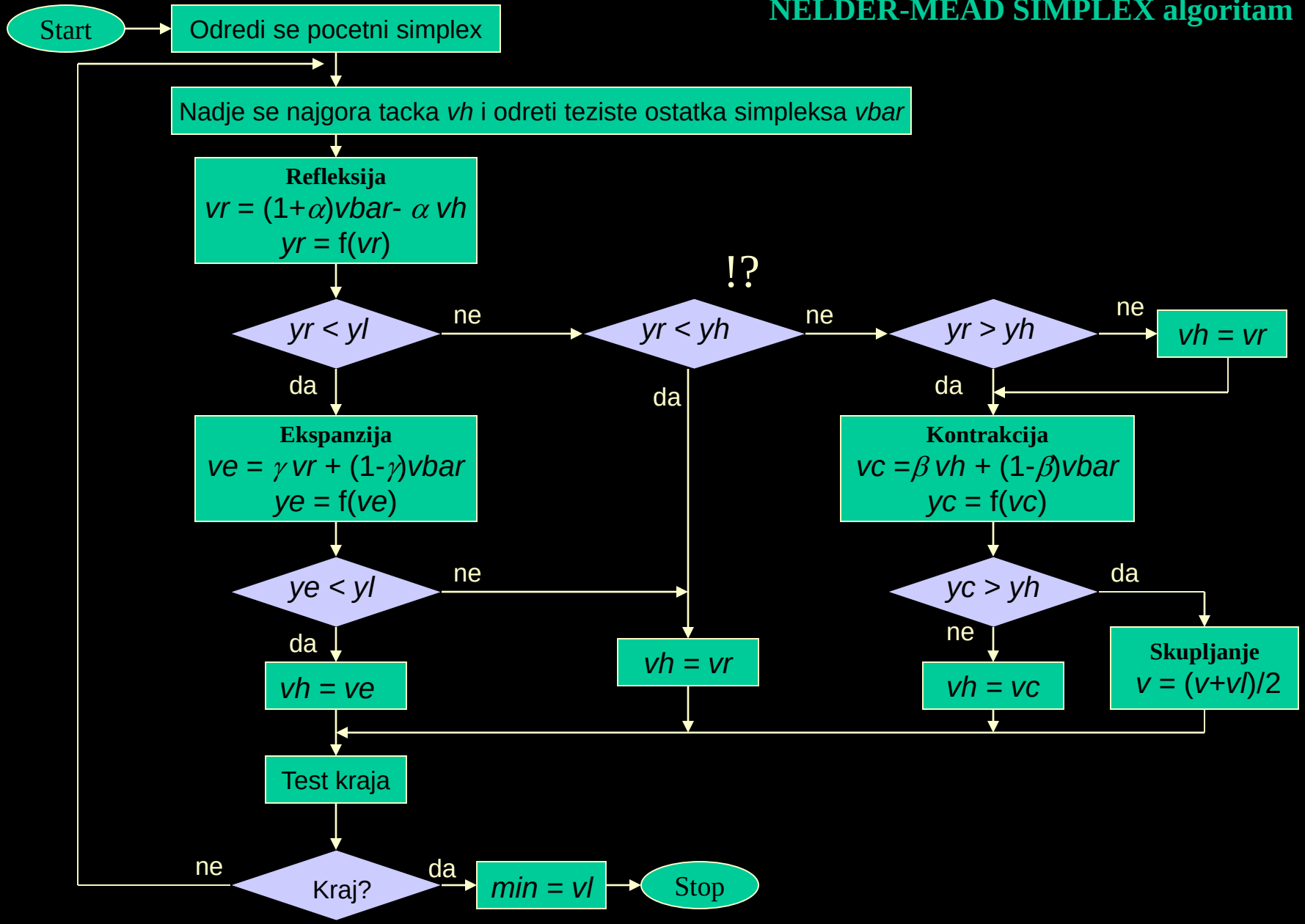
Nelder-Mead-ov *Simplex* algoritam

Metode optimizacije

NELDER-MEAD SIMPLEX algoritam



```
function [x,fx,cnt] = NeldMead(fun,x,xtol,ftol,maxit)
% Nelder-Mead alg. minimizacije f-je vise promenljivih bez ograničenja
%   fun - ime funkcije f(x)
%   x - početna tačka
%   xtol, ftol, maxit - kriterijumi zaustavljanja

n = prod(size(x));                  % broj promenljivih
maxit = maxit*n;                    % maksimlno dozvoljen broj računanja f-je

% Postavljanje početnog simplex-a
xin = x(:);                         % xin je vektor kolona
v = xin; fv = feval(fun,v);         % prvo teme i vrednost f-je u temenu

for j = 1:n                         % dodatnih N temena (ne smeju se poklapati)
   y = xin;
   if y(j) ~= 0
      y(j) = 1.05*y(j);             % pomeranje j-te koordinate za 5%
   else                             % ili
      y(j) = 0.00025;               % npr. 0.00025 (po preporuci iz lit.)
   end
   v = [v y];                       % dodaj teme (vertex)
   fv = [fv feval(fun,y)];          % i vrednost f-je u temenu
end                                 % dimenzija v=(n) x (n+1)
[fv,j] = sort(fv);                  % sortiranje temena prema vrednosti f-je u njima
v = v(:,j);                         % rasporedi temena: 1. je najniže,... N+1. najvišlje
cnt = n+1;                          % broj poziva f-je
```

```
alpha = 1;  beta = 1/2;  gamma = 2;
onesn = ones(1,n);
ot = 2:n+1;                          % sva temena bez najnižeg
on = 1:n;                            % sva temena, bez najvišljeg - koje treba pomeriti

% radi dok dijametar simplex-a ne bude manji od xtol ili razlika vrednosti f-je u temenima manja od ftol
% ili dok broj izračunavanja f-je ne premaši maxit
while cnt < maxit                                   % glavna petlja
    if max(max(abs(v(:,ot)-v(:,onesn)))) <= xtol & ...
           max(abs(fv(1)-fv(ot))) <= ftol
        break                                       % prekid petlje
    end

    % korak Nelder-Mead simplex algoritma
    vbar = (sum(v(:,on)')/n)';                      % težište (ne računa se najvišlje teme)
    vr = (1+alpha)*vbar - alpha*v(:,n+1);           % refleksija (uvek se proba)
    fr = feval(fun,vr);  cnt = cnt+1;
    vk = vr;  fk = fr;
    if fr < fv(n)                                   % nova tačka ispod do-najvišlje tačke?
        if fr < fv(1)                               % nova tačka je ispod najniže?
            ve = gamma*vr + (1-gamma)*vbar;                 % ekspanzija - proba
            fe = feval(fun,ve);  cnt = cnt+1;
            if fe < fv(1)                           % i posle ekspanzije imam još nižu tačku?
                vk = ve; fk = fe;                   % ekspanzija usvojena
            end
        end       % ako reflektovana tačka nije najniža, a niža je od do-najvišlje, usvaja se
```

```
    else                                        % reflektovana tačka nije bolja od do-najvišlje
        vt = v(:,n+1); ft = fv(n+1);            % najvišlja tačka (teme)
        if fr < ft                              % refl. tačka se poredi sa najvišljom ...
            vt = vr; ft = fr;                   % i za sada zamenjuje najvišlju
        end
        vc = beta*vt + (1-beta)*vbar;           % kontrakcija - proba
        fc = feval(fun,vc);  cnt = cnt+1;
        if fc < fv(n)                           % test kontrakcije
            vk = vc; fk = fc;                   % kontrakcija uspela
        else
            % ni refleksina ni kontrakcija ne daju nizu tačku od najvišlje
            for j = 2:n                         % polovljenje, najniže teme je i dalje 1.
                % moraju se izračunati nova temena simplex-a i vrednosti f-je u njima
                v(:,j) = (v(:,1) + v(:,j))/2;
                fv(j) = feval(fun,v(:,j));
            end
        vk = (v(:,1) + v(:,n+1))/2;             % najvišlje teme je izdvojeno zbog progr. rešenja
        fk = feval(fun,vk);  cnt = cnt+n;
        end
    end
    v(:,n+1) = vk;                              % zamena temena sa max. vrednosti
    fv(n+1) = fk;
    [fv,j] = sort(fv);                          % ponovo sortiranje
    v = v(:,j);
end                                             % kraj glavne petlje
```

```
x(:) = v(:,1);                                  % epilog
fx = fv(1);
if cnt == maxit
  disp(['Upozorenje: Maksimalan broj iteracija (', ...
              int2str(maxit),') je premasen']);
end
```