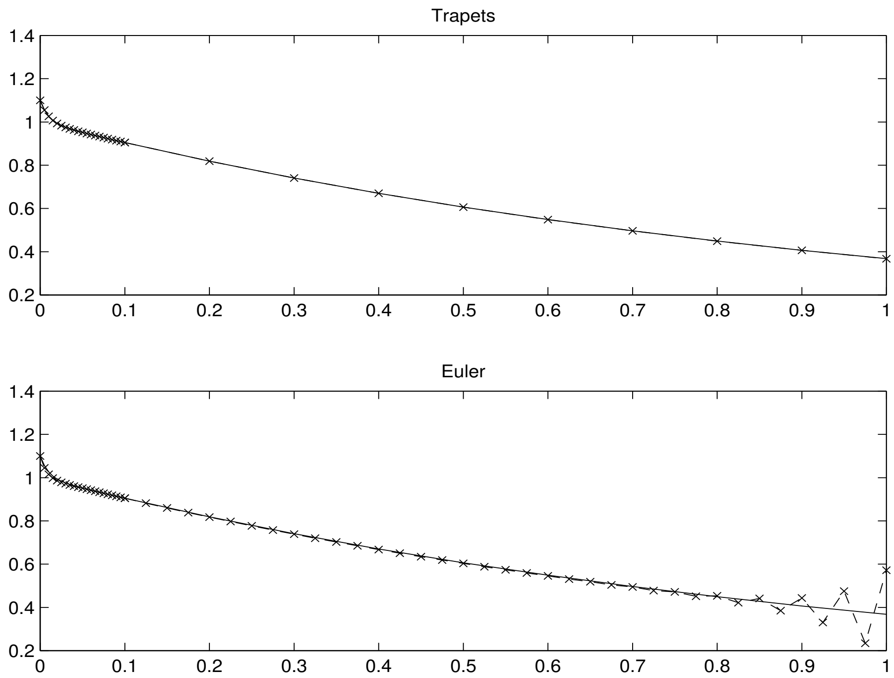
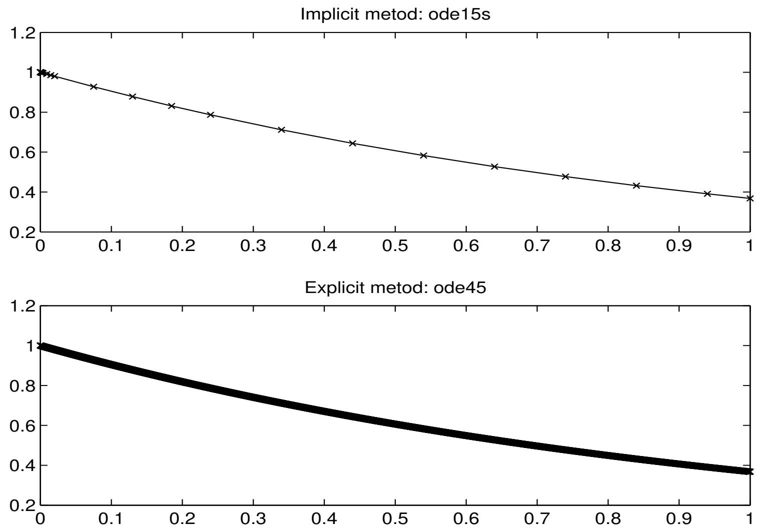


Figur 10.9: Trapetsmetoden (överst) och Eulers metod för ett styvt system. Punkterna där approximationer beräknats markeras med $\times$.

toden `ode45` och den implicita metoden `ode15s`, som är konstruerad för styva system. Båda metoderna har adaptiv steglängdskontroll, se Avsnitt 10.8, och försöker ta så långa steg som möjligt med hänsyn till stabilitet och noggrannhet (vi ställer samma krav på noggrannhet för båda metoderna). Resultaten illustreras i Figur 10.10. Den implicita metoden `ode15s` gör 99 funktionsevalueringar, medan den explicita gör 19129.

10.8 Runge-Kutta med adaptiv steglängdskontroll

När man löser ett begynnelsevärdesproblem för en ordinär differentialekvation är det önskvärt att kunna variera steglängden beroende på hur snabbt lösningen varierar. Eftersom lösningens utseende i regel inte är känd i förväg måste detta ske genom att man uppskattar det lokala trunkeringsfelet. Detta kan ske på följande sätt.



Figur 10.10: Lösning av ett styvt system med `ode15s` (övre grafen) och `ode45` (undre grafen; markeringarna ligger så tätt att de inte kan urskiljas).

Antag att vi använder en Runge-Kutta metod med globalt fel $O(h^4)$. En approximation y_n satisfierar då

$$y_n = y(x_n) + a(x_n)h^4 + O(h^5),$$

där den dominerande termen $a(x_n)$ är okänd. Om vi samtidigt har beräknat en approximation med en annan metod, som har globalt fel $O(h^5)$, så gäller för denna

$$\hat{y}_n = y(x_n) + O(h^5).$$

Vi kan då uppskatta det globala felet med $y_n - \hat{y}_n$. För att detta skall bli effektivt, måste man konstruera två metoder, *som amvänder samma* k_1, k_2, $k_3, \ldots$ I följande Matlabkod implementeras Runge-Kutta-Fehlbergs metod, som är av denna typ.

```
function [y,err]=rk45(ffun,x0,y0,h);
% Ett steg med R-K-Fehlberg45
A=[0 1/4 3/8 12/13 1 1/2];
B=[0 0 0 0 0 0
   1/4 0 0 0 0 0
   3/32 9/32 0 0 0 0
   1932/2197 -7200/2197 7296/2197 0 0 0
   439/216 -8 3680/513 -845/4104 0 0
   -8/27 2 -3544/2565 1859/4104 -11/40 0];
Y=[25/216 0 1408/2565 2197/4104 -1/5]';
YH=[16/135 0 6656/12825 28561/56430 -9/50 2/55]';
E=[1/360 0 -128/4275 -2197/75240 1/50 2/55]';

k1=h*feval(ffun,x0,y0);
k2=h*feval( ffun,x0+A(2)*h,y0+B(2,1)*k1);
k3=h*feval(ffun,x0+A(3)*h,y0+[k1 k2]*B(3,1:2)');
k4=h*feval(ffun,x0+A(4)*h,y0+[k1 k2 k3]*B(4,1:3)');
k5=h*feval(ffun,x0+A(5)*h,y0+[k1 k2 k3 k4]*B(5,1:4)');
k6=h*feval(ffun,x0+A(6)*h,y0+[k1 k2 k3 k4 k5]*B(6,1:5)');

y=y0+[k1 k2 k3 k4 k5]*Y;            % globalt fel=O(h^4)
yh= y0+[k1 k2 k3 k4 k5 k6]*YH;      % globalt fel=O(h^5)
err=norm([k1 k2 k3 k4 k5 k6]*E);    % feluppskattning
```

Denna metod implementeras i en adaptiv kod, som väljer steglängd beroende på uppskattningen av trunkeringsfelet: om felet är för stort i förhållande till en given tolerans, så görs steget om med mindre värde på h. Om det uppskattade felet är mycket mindre än den givna toleransen, så försöker man nästa gång med ett längre steg.

```
function [x,y,nfun]=adrk45(ffun,a,b,h0,y0,tol);
% Runge-Kutta med adaptiv steglängdskontroll
% Indata:
% ffun funktionen som definierar högerledet i y'=f(x,y)
%      OBS att y och f är vektorer
% a    startpunkt för integrationen
% b    slutpunkt för integrationen
% h0   initial steglängd
% y0   vektor av begynnelsevärden
% tol  tolerans för steglängdskontroll
% Utdata:
% x    vektor av x-värden där lösningen beräknats
% y    matris av funktionsvärden
% nfun antal funktionsevalueringar
%
i=0; nfun=0;
h=h0; xx=a+h; x0=a; y(:,1)=zeros(size(y0));
while xx<b,
  [yh,rt]=rk45(ffun,x0,y0,h);  % RK45 med steg h
  nfun=nfun+6;
  if rt<tol*h/(b-a),   % Acceptera detta steg
    i=i+1; x(i)=xx; y(:,i)=yh;
    if rt<0.05*tol*h/(b-a),    % längre steg nästa gång
      h=2*h;
    end
    x0=xx; y0=yh; xx=xx+h;
  else
    h=h/2; xx=x0+h;            % Försök med kortare steg
  end
end
i=i+1; h=b-x(i-1);             % Sista steget
y(:,i)=rk(ffun,x0,y0,h);
x(i)=b; nfun=nfun+6;
```

I Figur 10.11 redovisas en körning med `adrk45`. Differentialekvationen är den s k Brusselator-ekvationen, som är en modell för en kemisk reaktion,

$$y_1' \;=\; 1 + y_1^2 y_2 - 4y_1,$$

$$y_2' \quad = \quad 3y_1 - y_1^2 y_2,$$

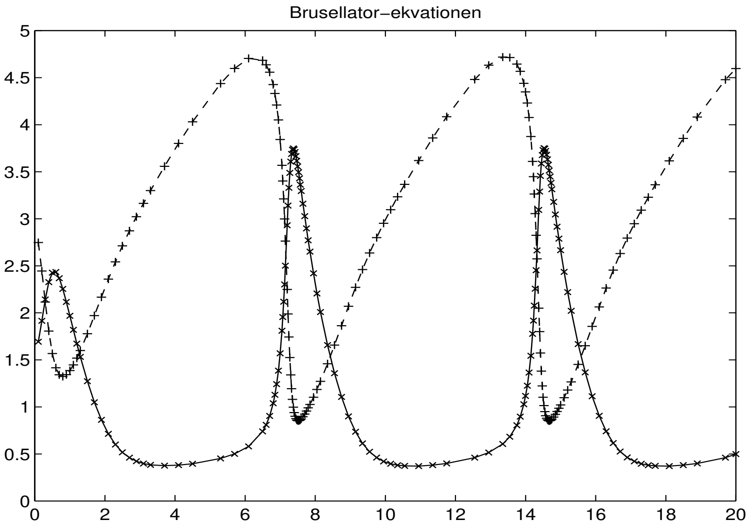
med begynnelsevillkoret $y_1(0) = 1.5,\ y_2(0) = 3$. Kod för denna ekvation ges av

```
function [f]=ff_brus(x,y)
f=[1+y(1)^2*y(2)-4*y(1);
   3*y(1)-y(1)^2*y(2)];
```

Anropet är

```
>>    [xx,yy,nfun]=adrk45(@ff_brus,0,20,0.1,[1.5; 3],0.001);
```

och resultatet illustreras i Figur 10.11.



Figur 10.11: R-K med automatisk steglängdskontroll. De beräknade punkterna markeras med $\times$ och $+$. 948 funktionsevalueringar utfördes, det längsta steget var 0.8 och det kortaste 0.025.