Numeričke metode za rešavanje Običnih Diferencijalnih Jednačina (ODJ)

Problem Početnih Vrednosti (PPV)

Šta su diferencijalne jedančine?

Diferencijalna jedančina je jedančina koja povezuje funkciju i njene izvode.

Red diferencijalne jedančine zavisi od reda izvoda. Diferencijalne jednačine prvog reda uključuju prve izvode, drugog reda druge, itd.

Izvod $\frac{dx}{dy}$ se naziva „običnim“ ukoliko je y funkcija samo jedne nezavisne promenljive x. U slučaju kada imamo funkciju $f(x, y)$ koja je funkcija više nezavisnih promenljivih x i y, izvodi $\frac{\partial f}{\partial x}$, $\frac{\partial f}{\partial y}$ se nazivaju „parcijalnim“ izvodima. Slično, imamo podelu na „obične“ i „parcijalne“ diferencijalne jednačine.

Obične diferencijalne jednačine sadrže jedan ili više običnih izvoda nepoznatih funkcija.

Diferencijalne jednačine se jako puno koriste za formiranje matematičkih modela različitih pojava u prirodi. Formule za mnoge zakone fizike su diferencijalne jednačine.

Na primer, Njutnov drugi zakon kretanja:

$$F = ma, \;\; a = \frac{dv}{dt}, \;\; v = \frac{dx}{dt}$$

gde F predstavlja silu, m masu, a ubrzanje, dv promenu brzine, dt promenu vremena, a dx promenu položaja. Rešavanjem sledeće jednačine dobijamo funkciju brzine po vremenu:

$$F(t, v) = m\frac{dt}{dv}$$

a rešavanjem sledeće jednačine dobijamo funkciju promene položaja po vremenu:

$$F(t, x, \frac{dx}{dt}) = m\frac{d^2x}{d^2t}$$

Ove jednačine su nam često od velike koristi kada nam je potrebno da uvedemo fiziku u video igre.

Na ovom predavanju bavićemo se običnim diferencijalnim jedančinama prvog reda, pa u nastavku dajemo nekoliko primera.

$$x + y' = e^x$$

$$\frac{dy}{dx} + 2y^2 + x = 1$$

$$y' = e^x$$

Početni uslovi

Ako uzmemo na primer diferencijalnu jedančinu:

$$y' = e^x$$

rešenja su sve funkcije oblika:

$$y(x) = e^x + C$$

gde je C proizvoljna konstanta. Ako želimo da imamo jedinstveno rešenje, tj. jednu funkciju onda moramo da zadamo neki uslov koji ta funkcija treba da zadovoljava. To se zove početni uslov. Na primer, za prethodni primer možemo da zadamo $y(0) = 1$ i tada dobijamo da je jedinstevno rešenje:

$$y(x) = e^x$$

Numeričke metode za rešavanje ODJ

Tejlorov red

Tejlorov red biće nam osnova za objašnjavanje svih metoda koje ćemo pokazati na ovom predavanju.

Tejlorov red za neku funkciju $f(x)$ u okolini neke tačke x_0 definisan je sa:

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{1}{2}f''(x_0)(x - x_0)^2 + \frac{1}{3!}f^{(3)}(x_0)(x - x_0)^3 + \dots$$

Ako uzmemo da je $x = x_0 + h$, Tejlorov red ima sledeći oblik:

$$f(x_0 + h) = f(x_0) + f'(x_0)h + \frac{1}{2}f''(x_0)h^2 + \frac{1}{3!}f^{(3)}(x_0)h^3 + \dots$$

Ovaj drugi oblik ćemo koristiti tokom predavanja jer će nam cilj biti da procenimo vrednosti funkcije $f(x)$ u nekoj tački $x_0 + h$ koja je za neki (obično mali) korak h udaljena od x_0. Tačka x_0 zadata nam je iz početnog uslova.

Ojlerov metod

Najejdnostavniji numerički metod za rešavanje ODJ.

Oslanja se na korišćenje prva dva člana Tejlorovog reda.

Pre nego što pokažemo metod, ponovićemo da nam je cilj da odredimo funkciju koja zadovoljava diferencijalnu jednačinu i početni uslov.

Formalnije, data nam je sledeća diferencijalna jedančina sa početnim uslovom:

$$F(x, y) = f(x, y), \;\; y(x_0)$$

Cilj nam je da nađemo funkciju $y(x_0)$ za koju važi jednačina $F(x,y)$ i koja u tački x_0 ima vrednost y_0.

Funkciju određujemo numerički. Polazimo od tačke $y(x_0)$ i koristeći neki od metoda određujemo tačke $y(x_0), y(x_1), y(x_2)\ldots$ koje predstavljaju aproksimaciju tačnih vrednosti funkcije $y(x)$ u tačkama $x_0, x_1, x_2\ldots$. Pogledaćemo sada jedan primer.

Data nam je diferencijalna jednačina:

$$f(x,y) = 2 - e^{-4x} - 2y,\ \ y(0) = 1$$

Rezultat primene jednog od numeričkih metoda za tačke $y(0), y(0.1), y(0.2)\ldots y(1)$ je:

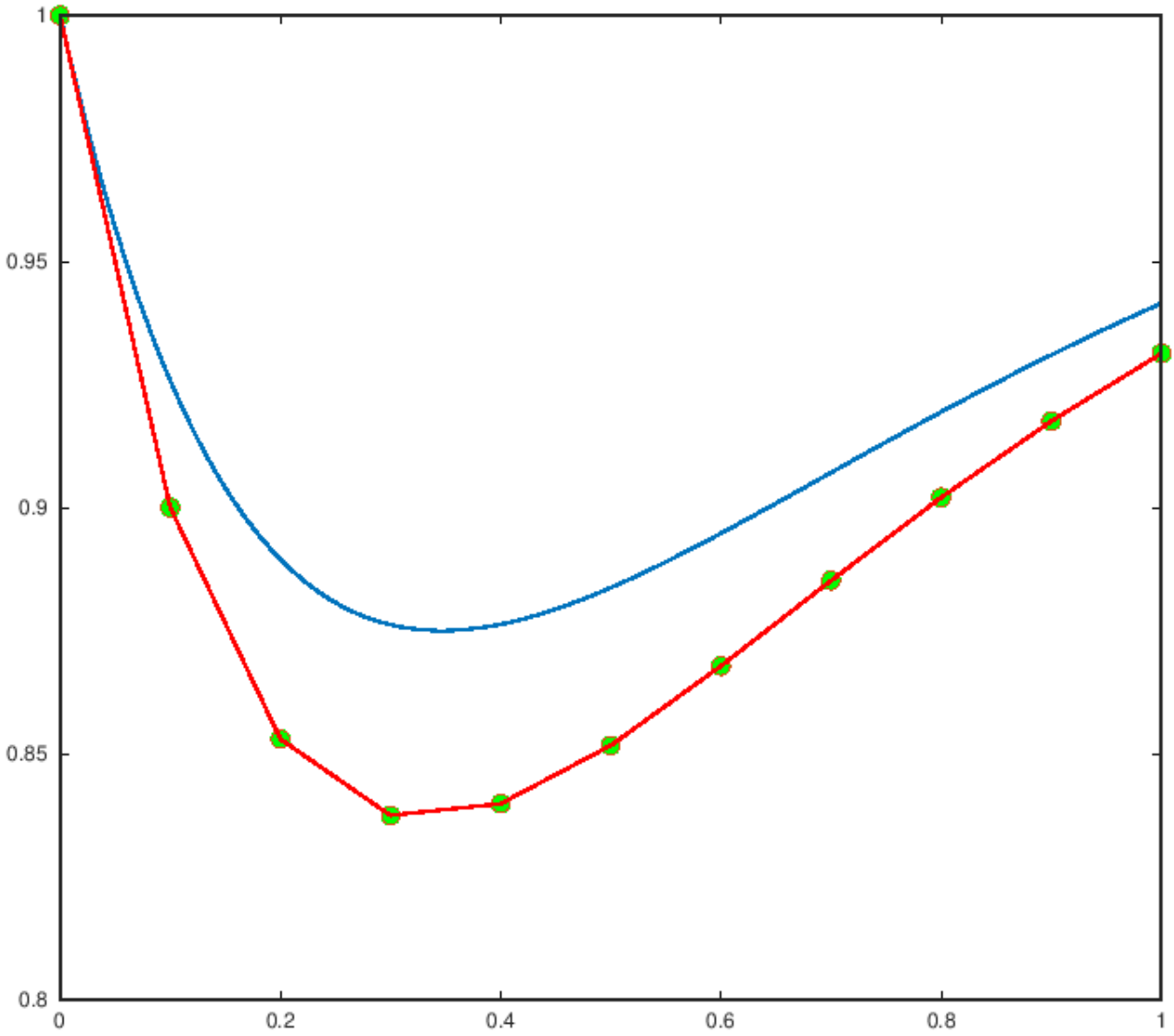
$$x = (0.00, 0.10, 0.20, 0.30, \ldots, 1.00)$$
$$y = (1.00, 0.90, 0.85, \ldots, 0.93)$$

Na sledećem grafiku prikazano je rešenje dobijeno numeričkim metodom (zelene tačke). Crvenom linijom nacrtan je polinom koji interpolira rešenje, dok je plavom linijom nacrtano tačno rešenje.

In [4]:
```
ode=@(x,y)2-exp(-4.*x)-2.*y;
fun=@(x,y)1+1/2.*exp(-4.*x)-1/2.*exp(-2.*x);
x0=0;y0=1;xn=1;h=0.1;
y=ojler(x0,xn,y0,h,ode)
```

```
y =

 Columns 1 through 8:

   1.00000   0.90000   0.85297   0.83744   0.83983   0.85168   0.86781   0.88517

 Columns 9 through 11:

   0.90206   0.91757   0.93132
```

In [5]:
```
plot_function([x0,xn],fun)
hold on;
xp=x0:0.1:xn;
plot(x0:h:xn,y,'o','markersize', 10,'markerfacecolor','g');
plot(xp,polyval(linterp(x0:h:xn,y),xp),"linewidth", 5,"color", "red");
```



Fromula za Ojlerov metod

Sada kada znamo na koji način, pomoću numeričkih metoda rešavamo ODJ, pokazaćemo kako funkcioniše Ojlerov metod.

Ojlerov metod oslanja se na prva dva člana Tejlorovog reda da aproksimira funkciju $y(x)$:

$$f(x_0 + h) = f(x_0) + f'(x_0)h + O(h^2) + \dots$$

Vidimo da nam za izračunavanje drugog člana treba prvi izvod funkcije. Na koji način ga dobijamo?

Upravo diferencijalna jednačina nam daje prvi izvod funkcije za koju god tačku želimo.

Ako u Tejlorov red zamenimo diferencijalnu jednačinu $f(x, y)$ dobijamo formulu za Ojlerov metod:

$$f(x_0 + h) = f(x_0) + hf(x_0, y_0) + O(h^2) + \dots$$

Ako sada sada uvedemo oznaku $y(x)$ umesto $f(x)$ i uvedemo oznaku za iteracije dobijamo sledeću formulu:

$$y(x_{i+1}) = y(x_i) + hf(x_i, y_i)$$

gde je $y(x)$ funkcija čija je diferencijalna jednačina data sa $f(x, y), y(x_0)$, a h proizvoljno odabrani korak.

Pre nego što primenimo Ojlerov metod na primer, objasnićemo ga i pomoću grafika.

Na slici ispod dat je prvi korak Ojlerovog metoda.

Sa slike se vidi da koristimo tangentu u početnoj tački $y(x_0)$. Za tangentu nam treba prvi izvod funkcije $y(x)$ u tački x_0. Njega dobijamo pomoću diferencijalne jedančine i iznosi $f(x_0, y_0)$. Koristimo onda tangentu da se pomerimo za korak h po njoj. Na taj način dobijamo sledeću tačku $y(x_1)$.

Na sledećoj slici dat je drugi korak metoda.

Sa slike se vidi da metod možemo da primenjujemo iterativno koliko god koraka želimo. Rezultat je skup vrednosti y kao na slici ispod. (Redni brojevi slika su iz udžbenika i možete ih ignorisati).

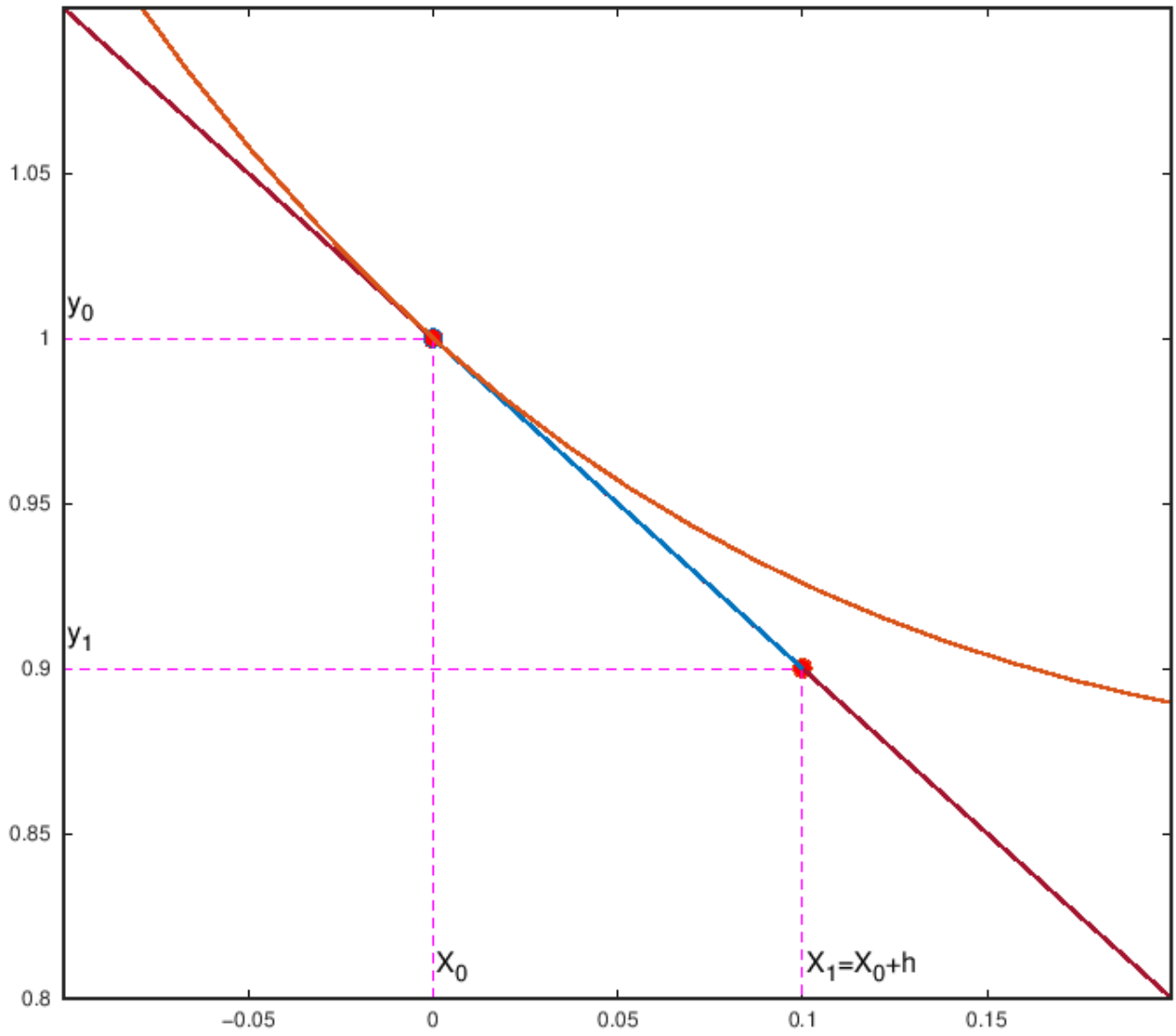
Primenjujemo sada Ojlerov metod na primer $f(x,y) = 2 - e^{-4x} - 2y,\ \ y(0) = 1$, **za** $h = 0.1$.

$$y(0.1) = y(0) + 0.1f(0,1) = 1 + 0.1 \cdot (2 - e^{-4 \cdot 0} - 2 \cdot 1) = 0.90$$

$$y(0.2) = y(0.1) + 0.1f(0.1,0.9) = 1 + 0.1 \cdot (2 - e^{-4 \cdot 0.1} - 2 \cdot 0.9) = 0.85$$

Prikazaćemo sada grafički jedan korak Ojlerovog metoda baš za ovaj primer, a nakon toga pišemo kod za Ojlerov metod.

In [6]:
```
ode=@(x,y)2-exp(-4.*x)-2.*y;
fun=@(x,y)1+1/2.*exp(-4.*x)-1/2.*exp(-2.*x);
x0=0;y0=1;xn=5;h=0.1;
plot_euler_st_1(x0,y0,h,ode)
plot_function([x0-0.1,xn],fun)
```



In [7]:
```
function y=ojler(x0,xn,y0,h,odj)
    x = x0:h:xn;
    n = length(x);
    y = zeros(1,n);
    y(1) = y0;
    for i=2:n
        y(i) = y(i-1) + h*feval(odj, x(i-1), y(i-1));
    end
endfunction
```

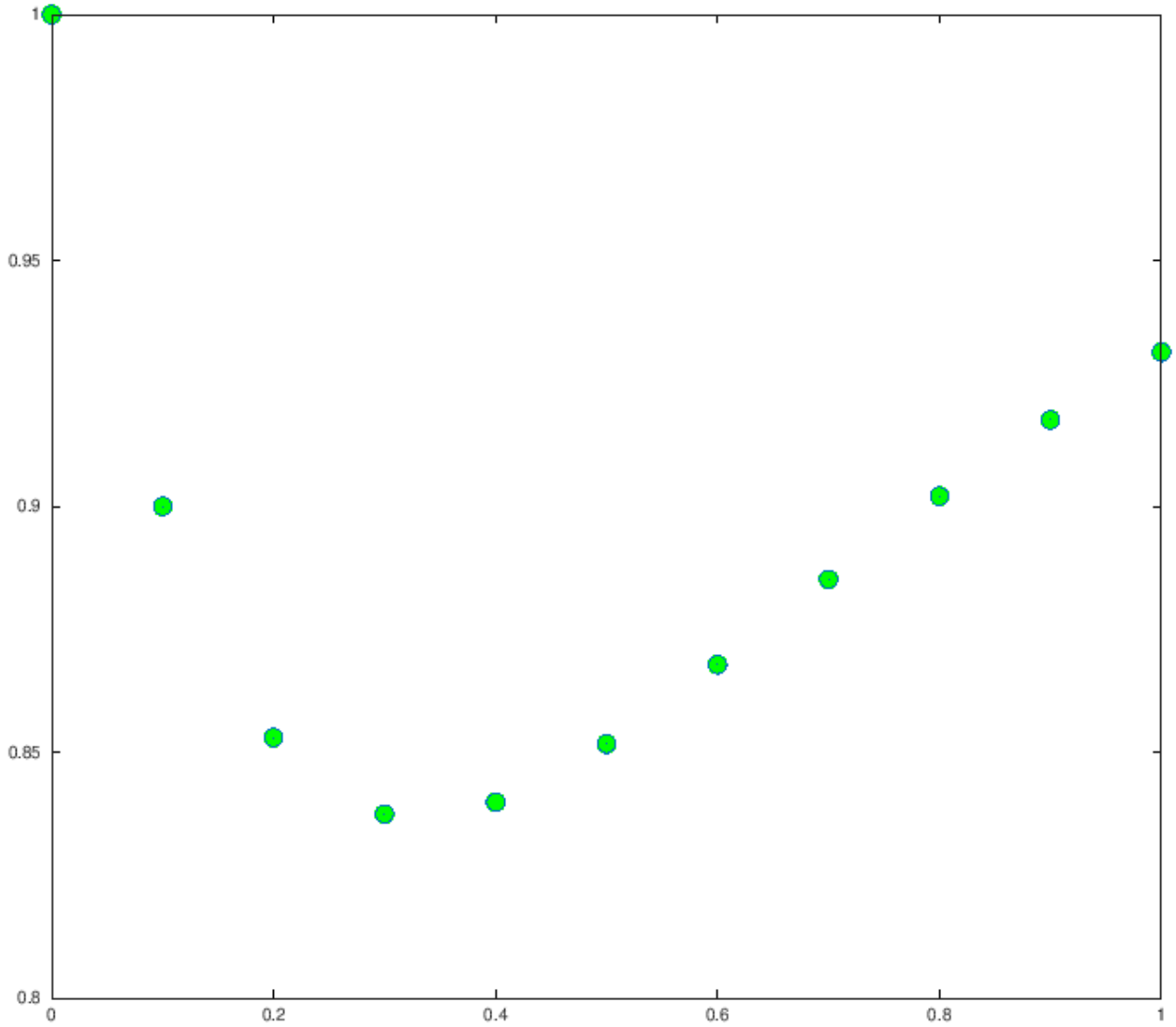
Pozivamo funkciju za primer $f(x,y) = 2 - e^{-4x} - 2y,\ \ y(0) = 1$, za $h = 0.1$.

In [8]:
```
ode=@(x,y)2-exp(-4.*x)-2.*y;
x0=0;y0=1;xn=1;h=0.1;
y=ojler(x0,xn,y0,h,ode)
y_ojler=y; %za kasnije poređenje
```

```
y =

 Columns 1 through 8:

   1.00000   0.90000   0.85297   0.83744   0.83983   0.85168   0.86781   0.88517

 Columns 9 through 11:

   0.90206   0.91757   0.93132
```

Prikazujemo tačke koje smo dobili.

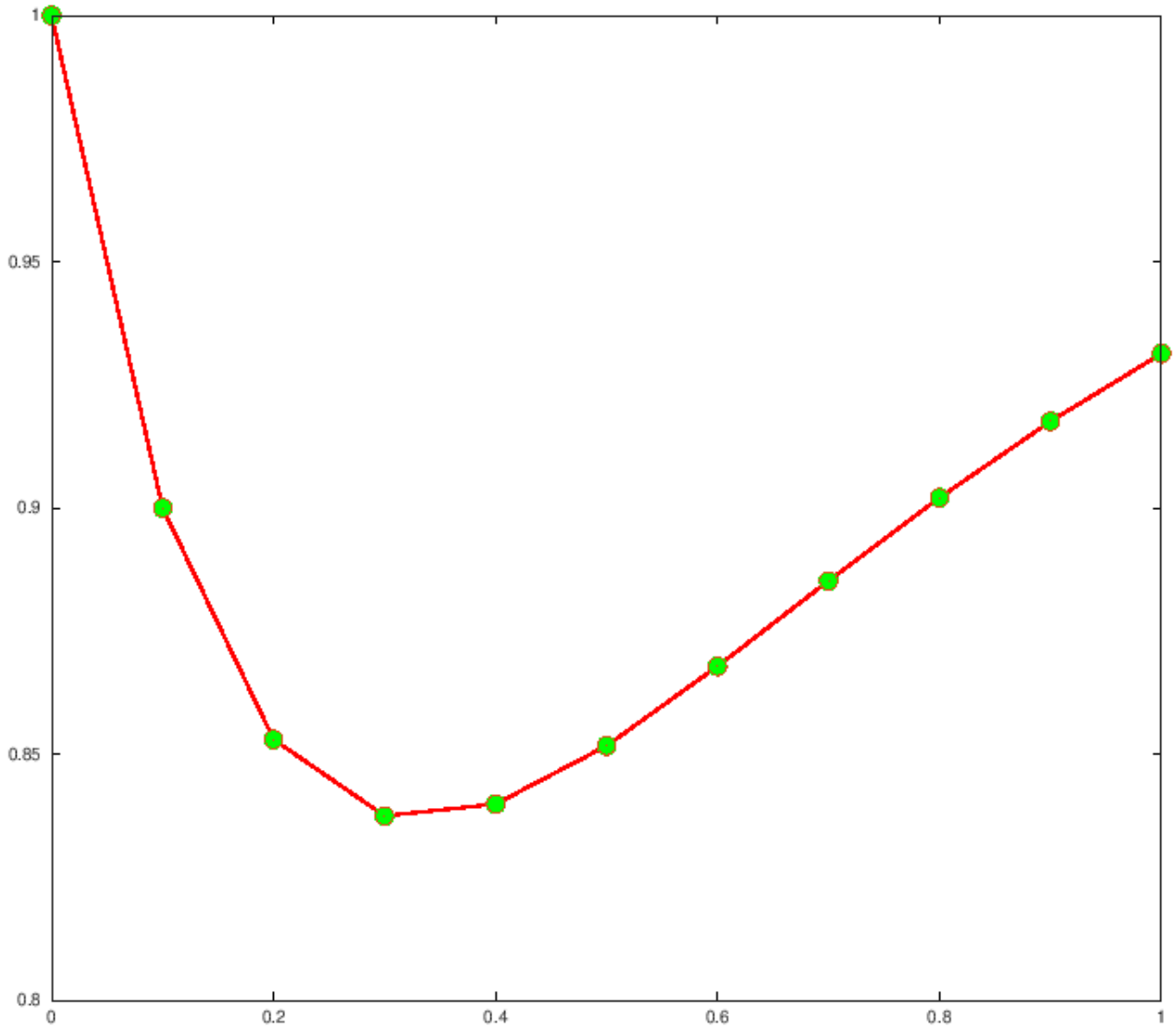
In [9]:
```
plot(x0:h:xn,y,'o','markersize', 10,'markerfacecolor','g')
```



Interpoliramo tačke i prikazujemo polinom.

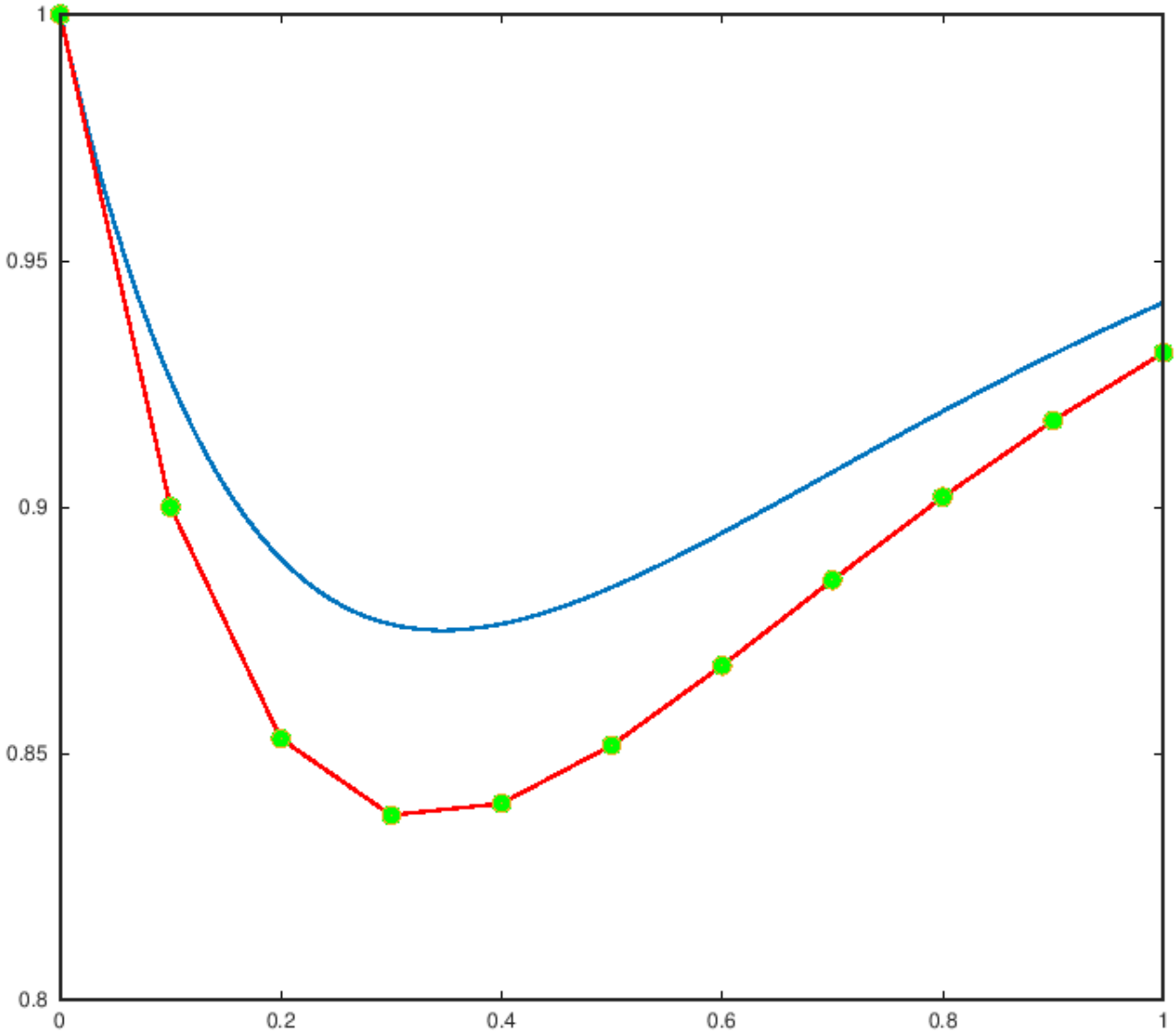
In [10]:
```
p=linterp(x0:h:xn,y)
plot(xp,polyval(p,xp),"linewidth", 5,"color", "red");
hold on;
plot(x0:h:xn,y,'o','markersize', 10,'markerfacecolor','g');
```

```
p =

 Columns 1 through 7:

   0.032015  -0.240746   0.888578  -2.192091   4.110311  -6.172883   7.416443

 Columns 8 through 11:

  -6.795557   4.249041  -1.363787   1.000000
```



Prikazujemo sada i tačno rešenje $y(x) = 1 + \frac{1}{2}e^{-4x} - +\frac{1}{2}e^{-4x}$ plavom bojom.

In [11]:
```
plot_function([x0,xn],fun);
hold on;
plot(xp,polyval(p,xp),"linewidth", 5,"color", "red");
plot(x0:h:xn,y,'o','markersize', 10,'markerfacecolor','g');
```



Vidimo da se rezultat Ojlerovog metoda razlikuje od tačnog rešenja, a u nastavku detaljnije pričamo grešci Ojlerovog metoda.

Greška Ojlerovog metoda

Pošto Ojlerov metod koristi prva dva člana Tejlorovog reda:

$$f(x_0 + h) = f(x_0) + hf(x_0, y_0) + O(h^2) + \dots$$

jasno je da je greška jedne primene Ojlerovog metoda reda:

$$O(h^2)$$

Greška jedne primene se često zove lokalna greška.

Ako metod primenimo n puta za neki korak h od tačke x_0 do tačke $x_n = x_0 + n \cdot h$ onda važi

$$n = \frac{x_n - x_0}{h}$$

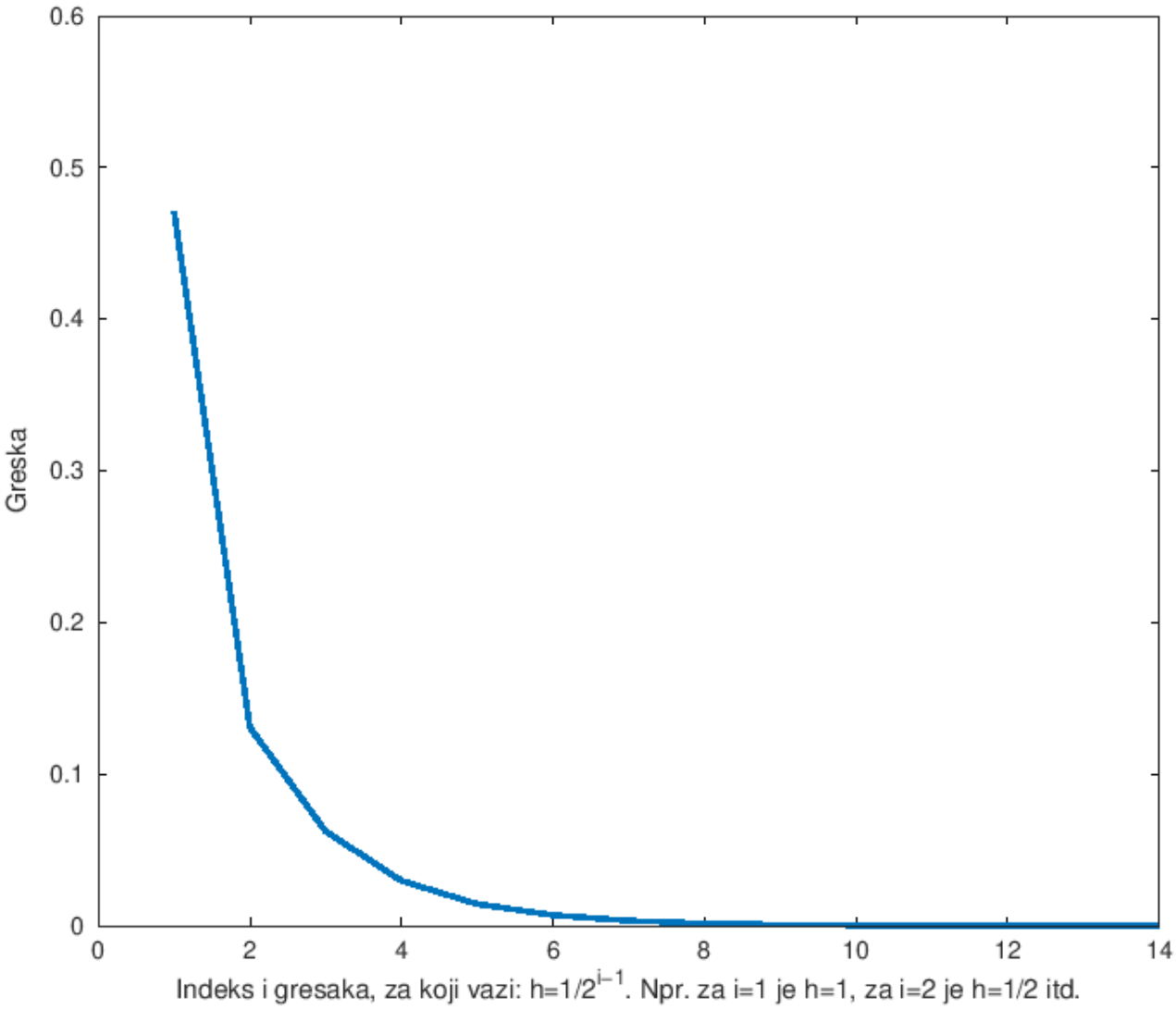
što je proprocionalno sa $O(\frac{1}{h})$. Grubo gledano (dokaz postoji u udžbeniku) zato je red greške n primena metode:

$$O(h)$$

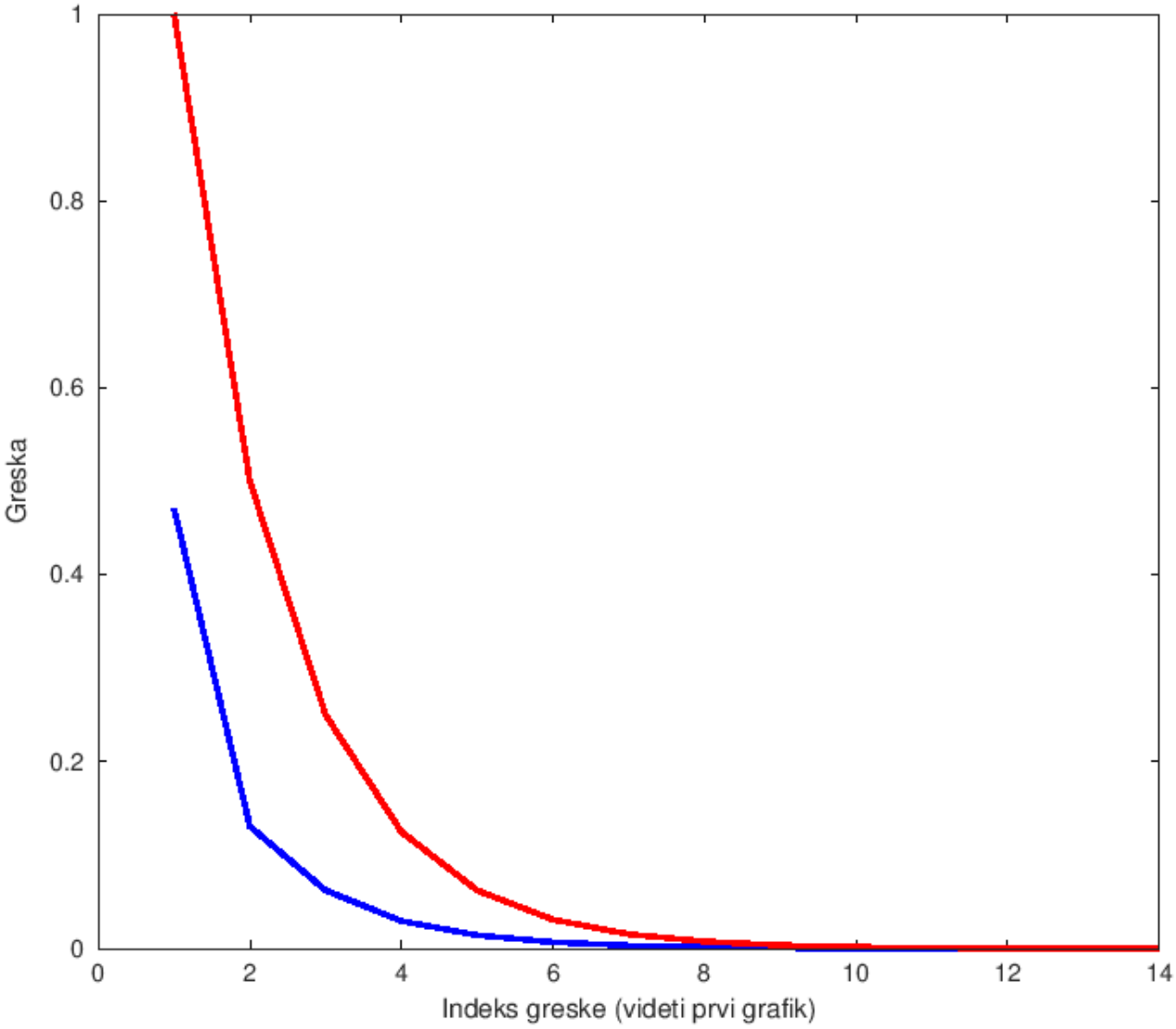
Red greške više primena metode često se zove globalni red greške.

U nastavku prikazujemo grešku Ojlerovog metoda za primer $f(x, y) = 2 - e^{-4x} - 2y, \;\; y(0) = 1$. Krećemo sa korakom $h = 1$ i smanjujemo ga tako što ga delimo sa 2 dok ne dođemo do koraka $h = 0.0001$. Nakon toga poredimo grafik greške i funkcije $f(h) = h$.

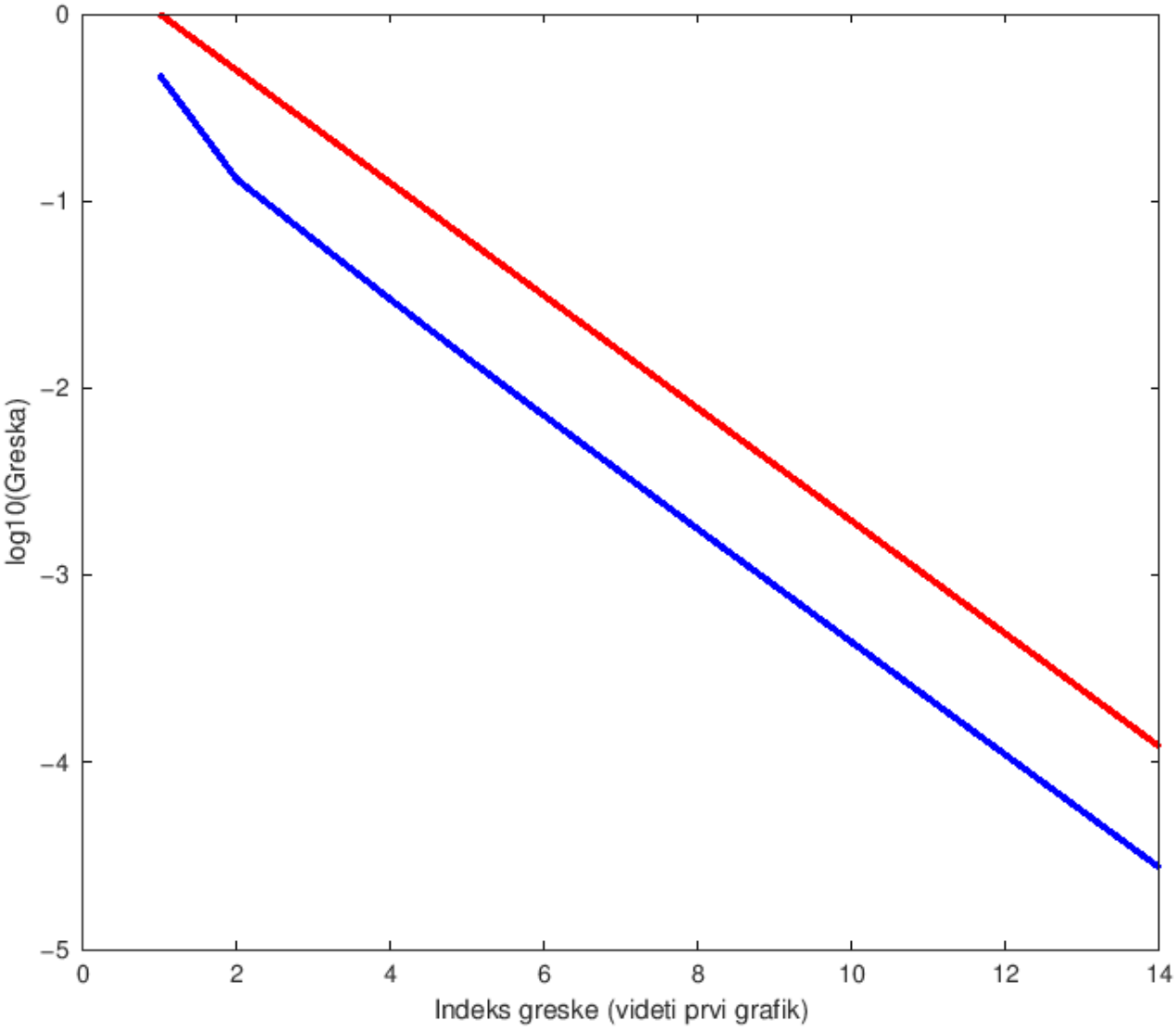
In [34]:
```
ode=@(x,y)2-exp(-4.*x)-2.*y;
x0=0;y0=1;xn=1;h=0.1;
[errors,sub_intervals]=calculate_error(x0,xn,y0,ode,fun,0.0001,'ojler');
errors_ojler=errors;
```



In [38]:
```
plot(1:length(errors),errors,"linewidth",10,"color","blue")
hold on;
plot(1:length(sub_intervals),sub_intervals.^1,"linewidth",10,"color", "red")
xlabel('Indeks greske (videti prvi grafik)');
ylabel('Greska');
set(gca, "fontsize", 14)
```



In [39]:
```
plot(1:length(errors),log10(errors),"linewidth",10,"color","blue")
hold on;
plot(1:length(sub_intervals),log10(sub_intervals.^1),"linewidth",10,"color", "red")
xlabel('Indeks greske (videti prvi grafik)');
ylabel('log10(Greska)');
set(gca, "fontsize", 14)
```



Sa prethodnih grafika može se videti da ako preplovimo korak h i greška se takođe prepolovi.

Videli smo da Ojlerov metod koristi prva dva člana Tejlorovog reda i ima relativno malu tačnost. Logično pitanje bilo bi zašto ne koristimo više članova Tejlorovog reda.

Problem je u tome što nam za te članove trebaju redom drugi, treći, četvrti... izvod funkcije, a oni nam nisu dati.

U nastavku pokazujemo prvo dva metoda koji daju tačnost koja je jednaka metodama koje koriste i treći član Tejlorovog reda. Nakon toga objašnjavamo na koji način smo postigli takvu tačnost bez korišćenja drugog izvoda.

Metod srednje tačke

Intuitivno, ovaj metod koristi tangentu u tački $x_0 + \frac{1}{2}h$ koja je na polovini intervala $[x_0, x_1]$ da stigne do tačke y_1. Ideja je u tome da je nagib te tangente bolja procena kretanja funkcije od tangente u početnoj tački x_0.

Formula za metod srednje tačke je:

$$x_{i\,\frac{1}{2}} = x_i + \frac{1}{2}h$$

$$y_{i\,\frac{1}{2}} = y_i + \frac{1}{2}hf(x_i, y_i)$$

$$y_{i+1} = y(x_i) + hf(x_{i\,\frac{1}{2}}, y_{i\,\frac{1}{2}})$$

gde je $y(x)$ funkcija koja čija je diferencijalna jednačina data sa $f(x, y), y(x_0)$, a h proizvoljno odabrani korak.

Pre nego što primenimo Metod srednje tačke na primer, objasnićemo ga i pomoću grafika.

Na slici ispod dat je prvi korak Metoda srednje tačke.

Primenjujemo sada Metod srednje tačke na primer $f(x, y) = 2 - e^{-4x} - 2y,\ \ y(0) = 1$, za $h = 0.1$.

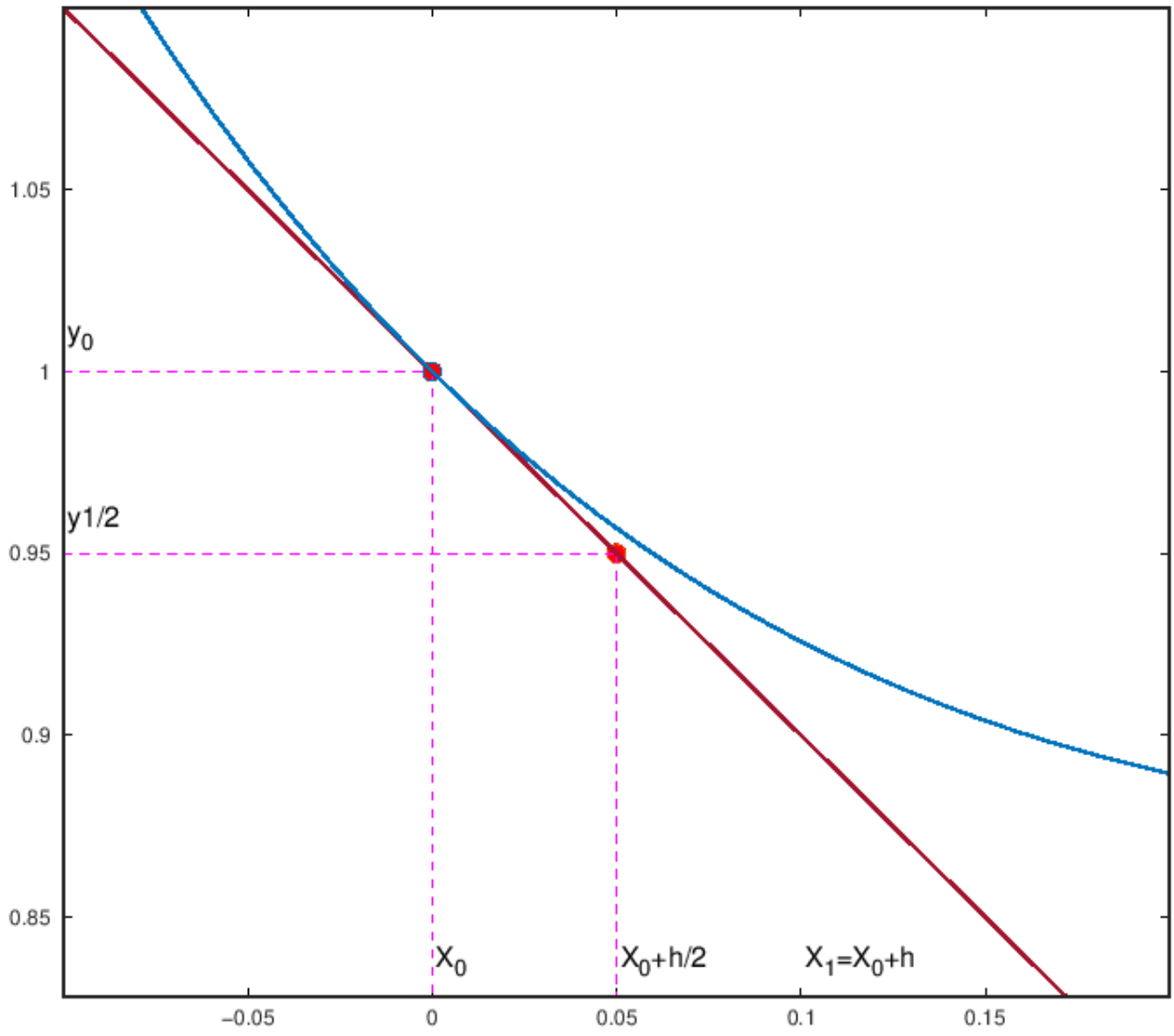
$$x_{1\,\frac{1}{2}} = x_0 + \frac{1}{2}0.1 = 0.05$$

$$y_{1\,\frac{1}{2}} = y(x_0 + \frac{1}{2}0.1) = y(0.05) = y(0) + 0.05f(0, 1) = 1 + 0.05 \cdot (2 - e^{-4 \cdot 0} - 2 \cdot 1) = 0.95$$

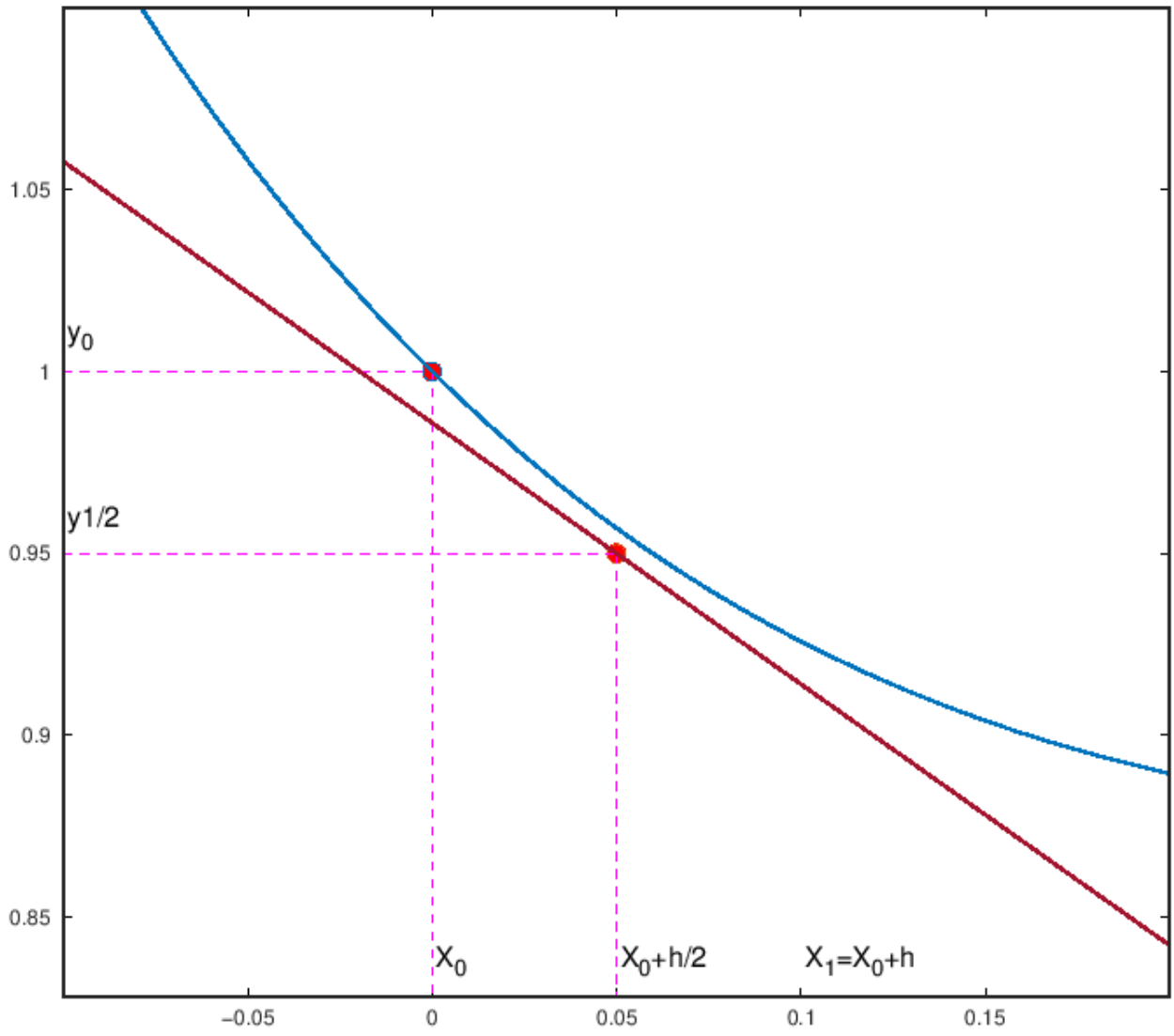
$$y(0.1) = y(0) + 0.1f(0.05, 0.95) = 1 + 0.1 \cdot (2 - e^{-4 \cdot 0.1} - 2 \cdot 0.95) = 0.92$$

Prikazaćemo sada grafički jedan korak metoda srednje tačke baš za ovaj primer, a nakon toga pišemo kod za metod srednje tačke.

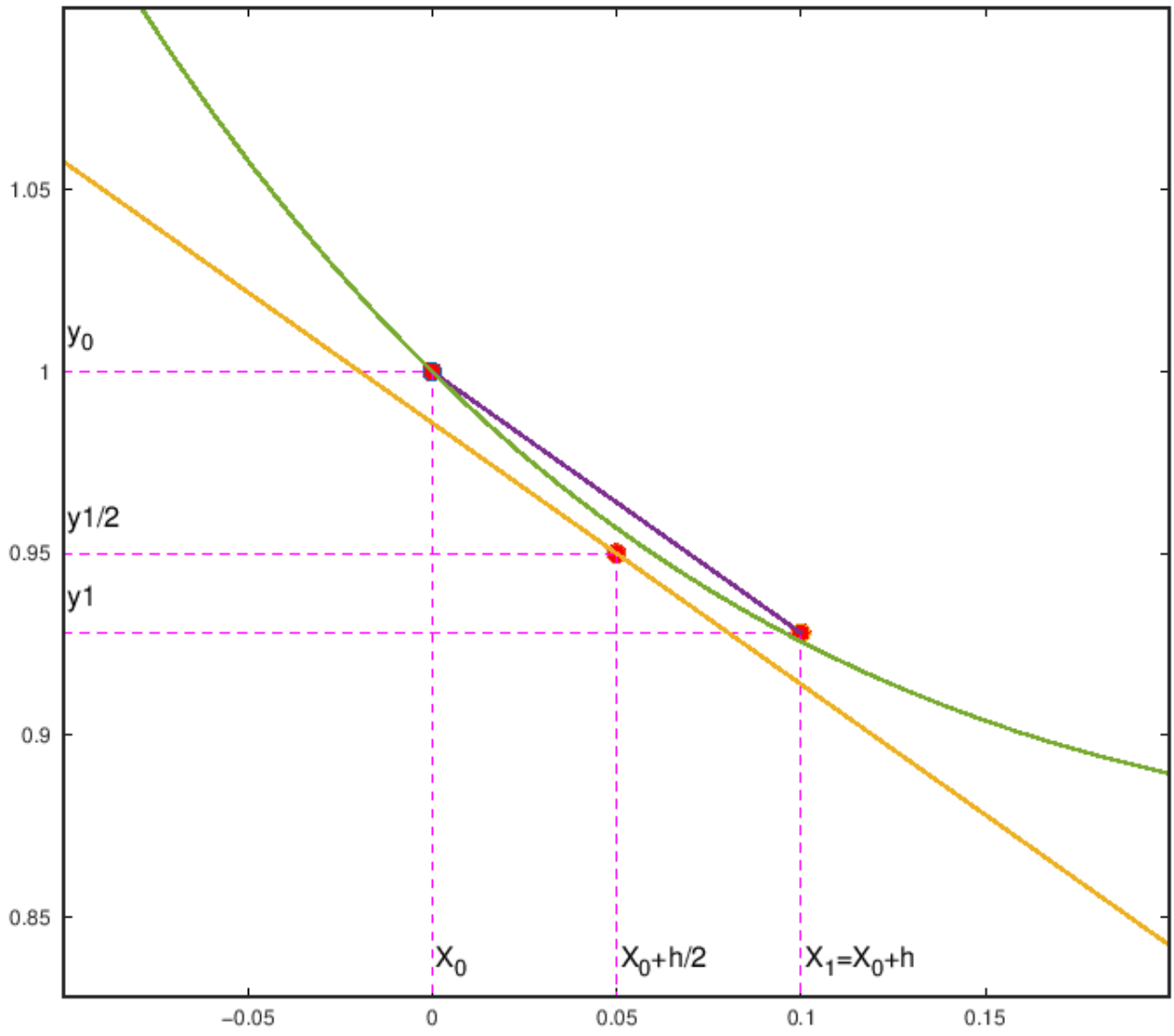
In [16]:
```
ode=@(x,y)2-exp(-4.*x)-2.*y;
fun=@(x,y)1+1/2.*exp(-4.*x)-1/2.*exp(-2.*x);
x0=0;y0=1;xn=3;h=0.1;
plot_mp_st_1(x0,y0,h,ode)
plot_function([x0-0.1,xn],fun)
```



In [17]:
```
plot_mp_st_2(x0,y0,h,ode)
plot_function([x0-0.1,xn],fun)
```



In [18]:
```
plot_mp_st_3(x0,y0,h,ode)
plot_function([x0-0.1,xn],fun)
```



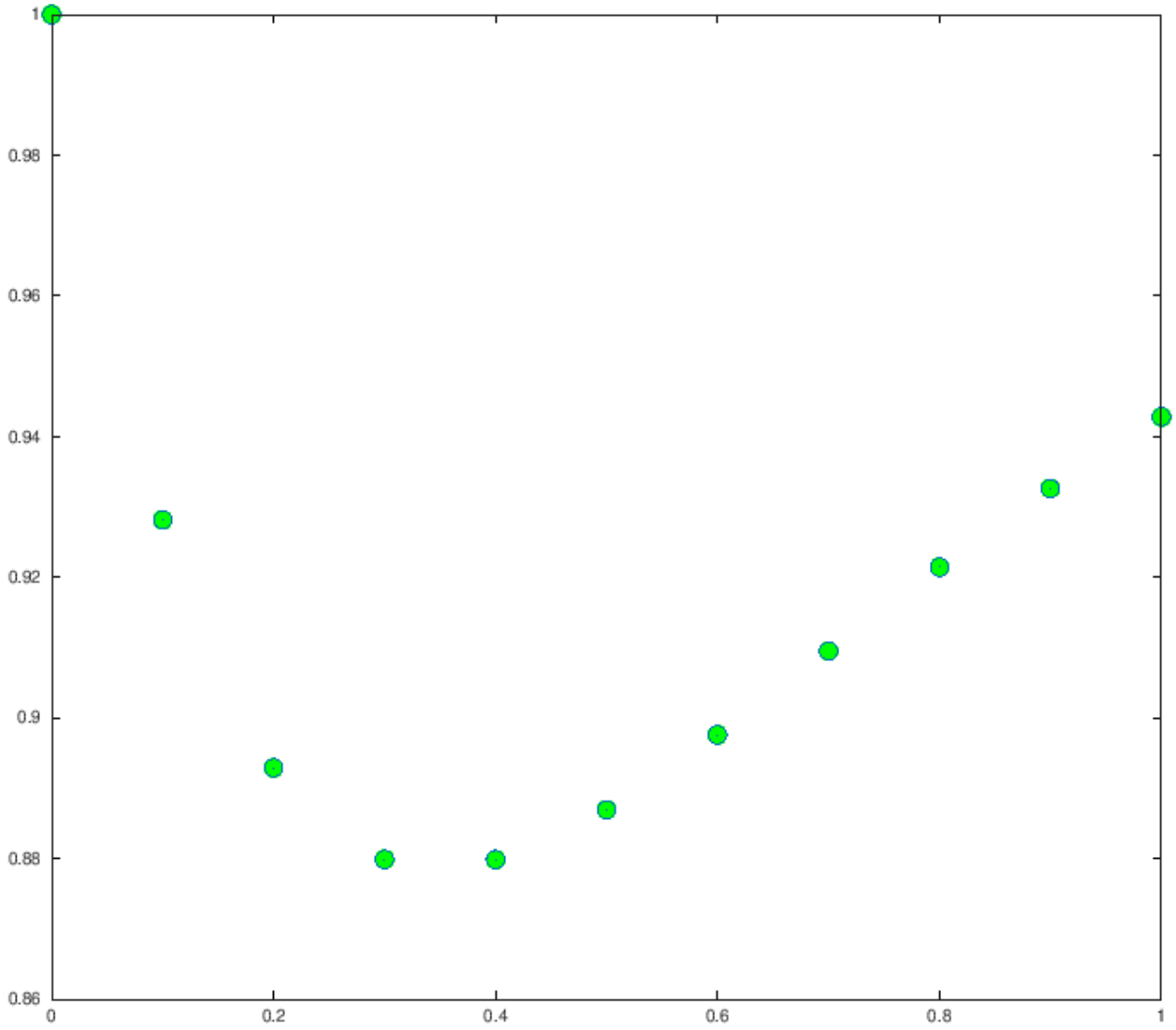
Pišemo sada kod za metod srednje tačke. Ulazni parametri su jednaki kao kod Ojlerovog metoda.

In [19]:
```
function y=stacka(x0,xn,y0,h,odj)
    x = x0:h:xn;
    n = length(x);
    y = zeros(1,n);
    y(1) = y0;
    for i=2:n
        x1_2 = x(i-1) + h/2;
        y1_2 = y(i-1) + h/2*feval(odj, x(i-1), y(i-1));
        y(i) = y(i-1) + h*feval(odj, x1_2, y1_2);
    end
endfunction
```

In [20]:
```
ode=@(x,y)2-exp(-4.*x)-2.*y;
x0=0;y0=1;xn=1;h=0.1;
y=stacka(x0,xn,y0,h,ode)
y_stacka=y;
```

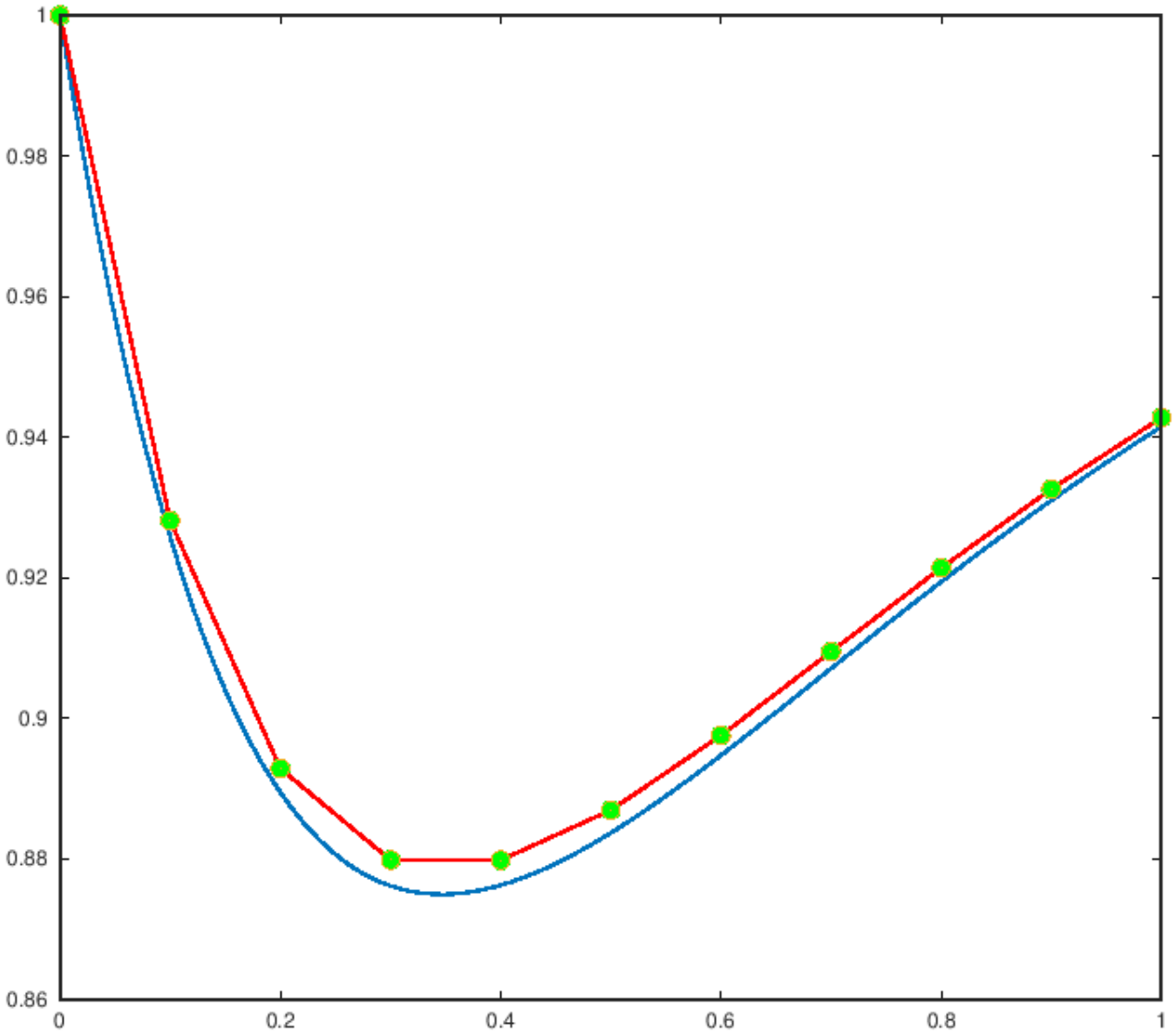
```
y =

 Columns 1 through 8:

   1.00000   0.92813   0.89289   0.87987   0.87985   0.88696   0.89758   0.90950

 Columns 9 through 11:

   0.92142   0.93263   0.94280
```

In [21]:
```
plot(x0:h:xn,y,'o','markersize', 10,'markerfacecolor','g');
```



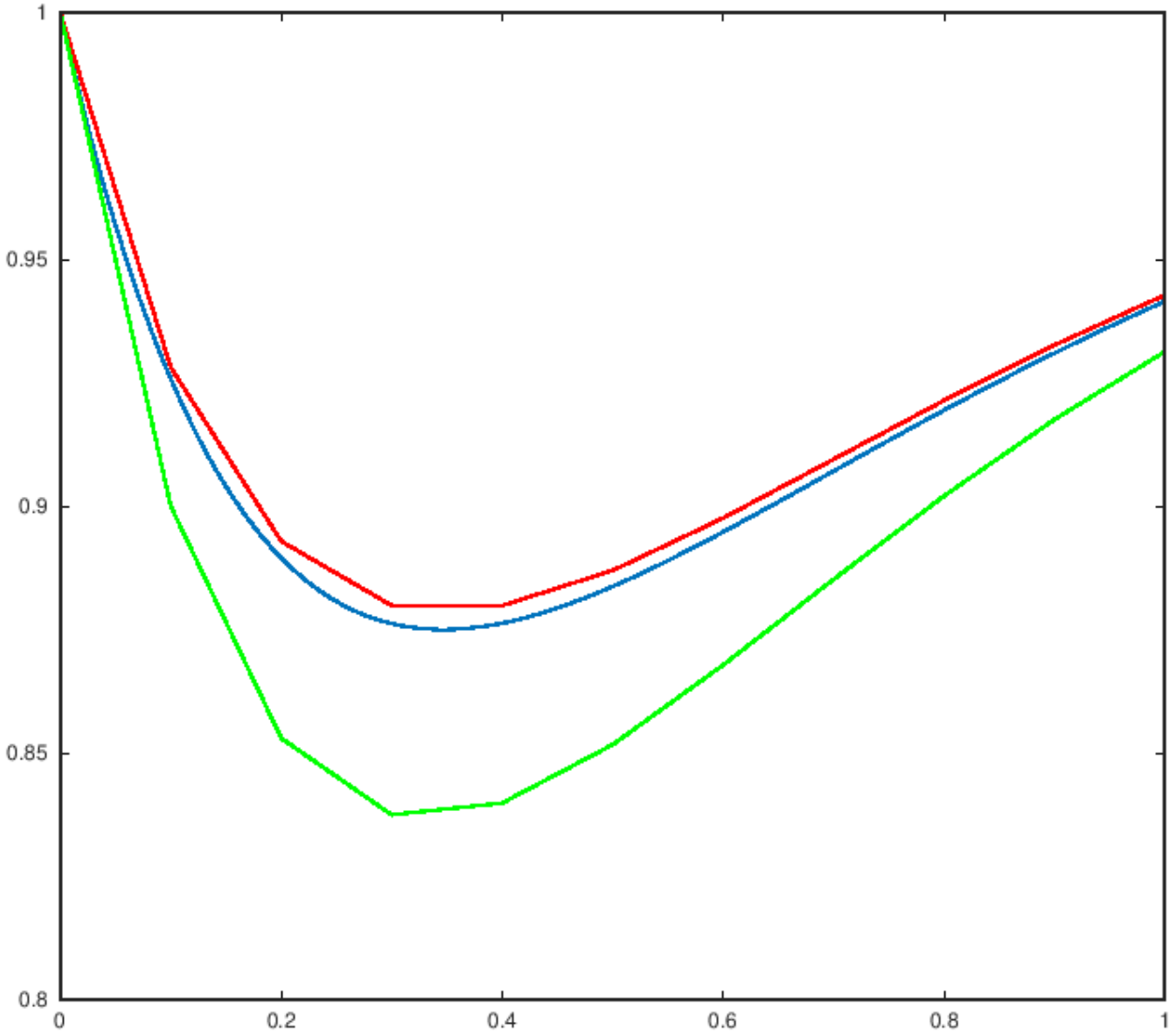
In [22]:
```
p=linterp(x0:h:xn,y)
plot_function([x0,xn],fun);
hold on;
plot(xp,polyval(p,xp),"linewidth", 5,"color", "red");
plot(x0:h:xn,y,'o','markersize', 10,'markerfacecolor','g');
```

```
p =

 Columns 1 through 7:

   0.020024  -0.150723   0.557311  -1.379328   2.600863  -3.941966   4.803902

 Columns 8 through 11:

  -4.495291   2.895796  -0.967792   1.000000
```



Poredimo Ojlerov metod (označen zeleno) i metod srednje tačke (označen crveno).

In [23]:
```
plot_function([x0,xn],fun);
hold on;
p_ojler=linterp(x0:h:xn,y_ojler);
plot(xp,polyval(p,xp),"linewidth", 5,"color", "red");
plot(xp,polyval(p_ojler,xp),"linewidth", 5,"color", "green");
```

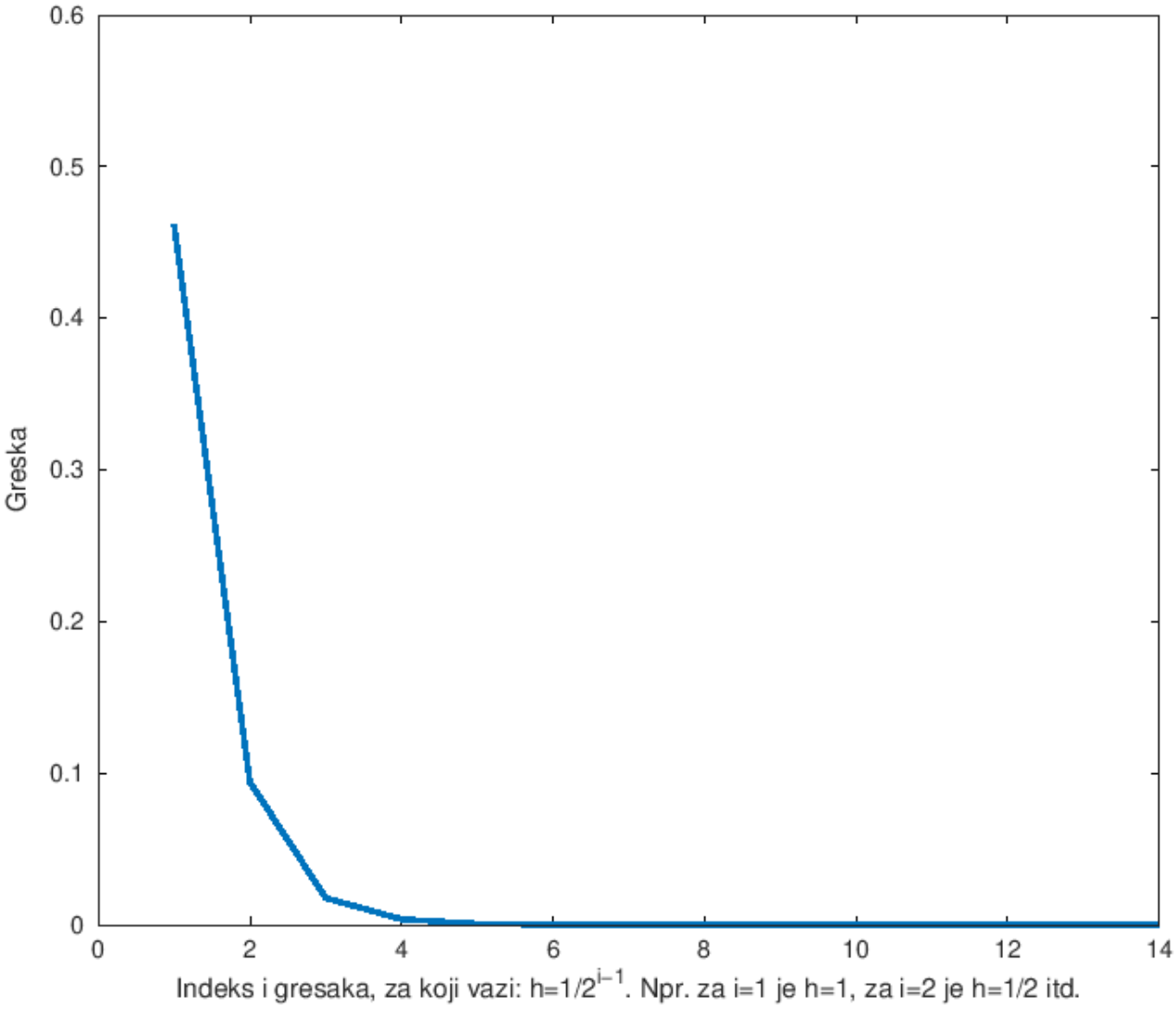


Vidimo da metod srednje tačke ima veću tačnost. Formalno tačnost metoda srednje tačke je reda:

$$O(h^2)$$

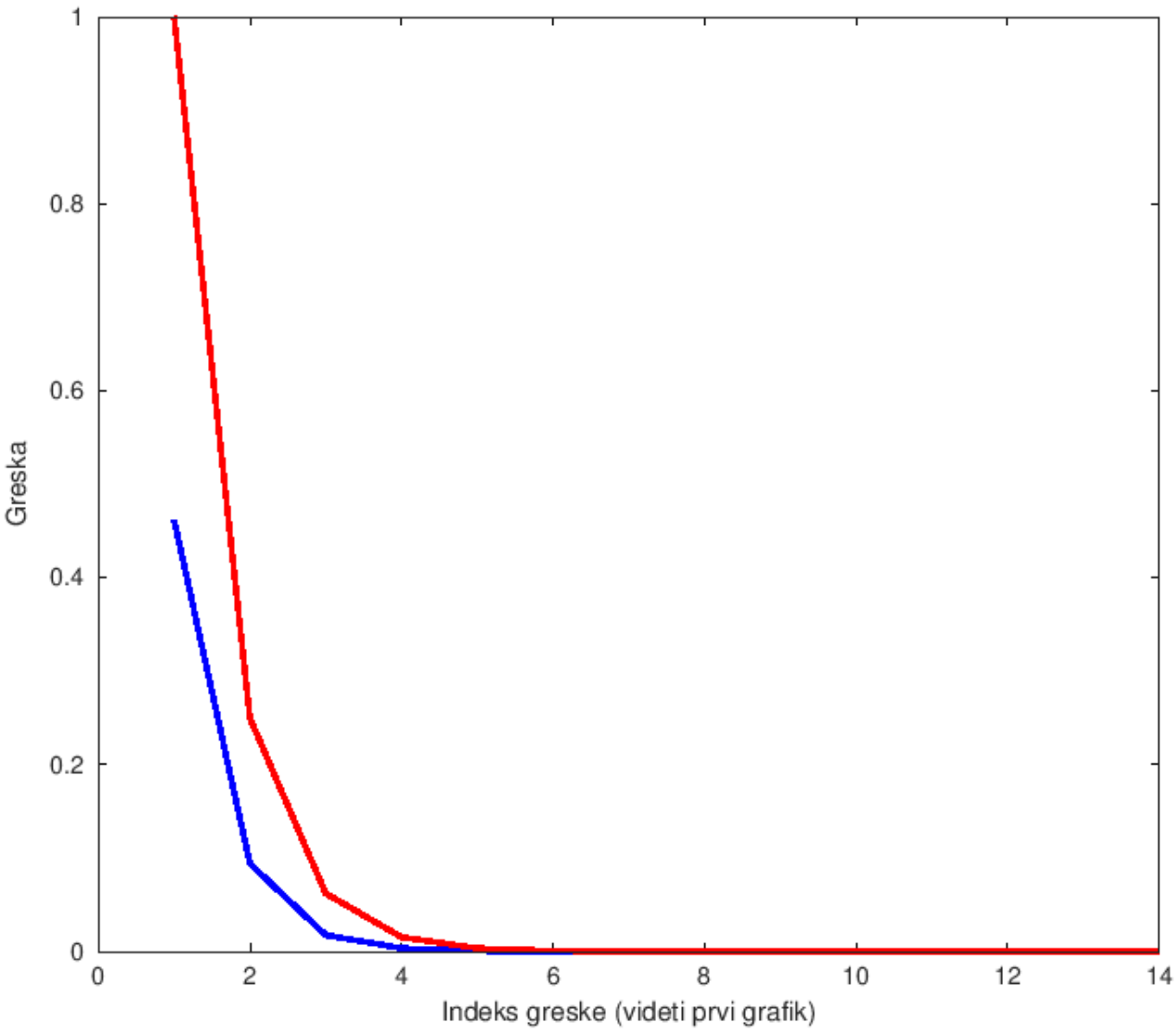
Kasnije objašnjavamo zato red greške ima baš ovu vrednost. Pokazaćemo sada grafike greške za metod srednje tačke.

In [40]:
```
[errors,sub_intervals]=calculate_error(x0,xn,y0,ode,fun,0.0001,'stacka');
errors_stacka=errors;
```

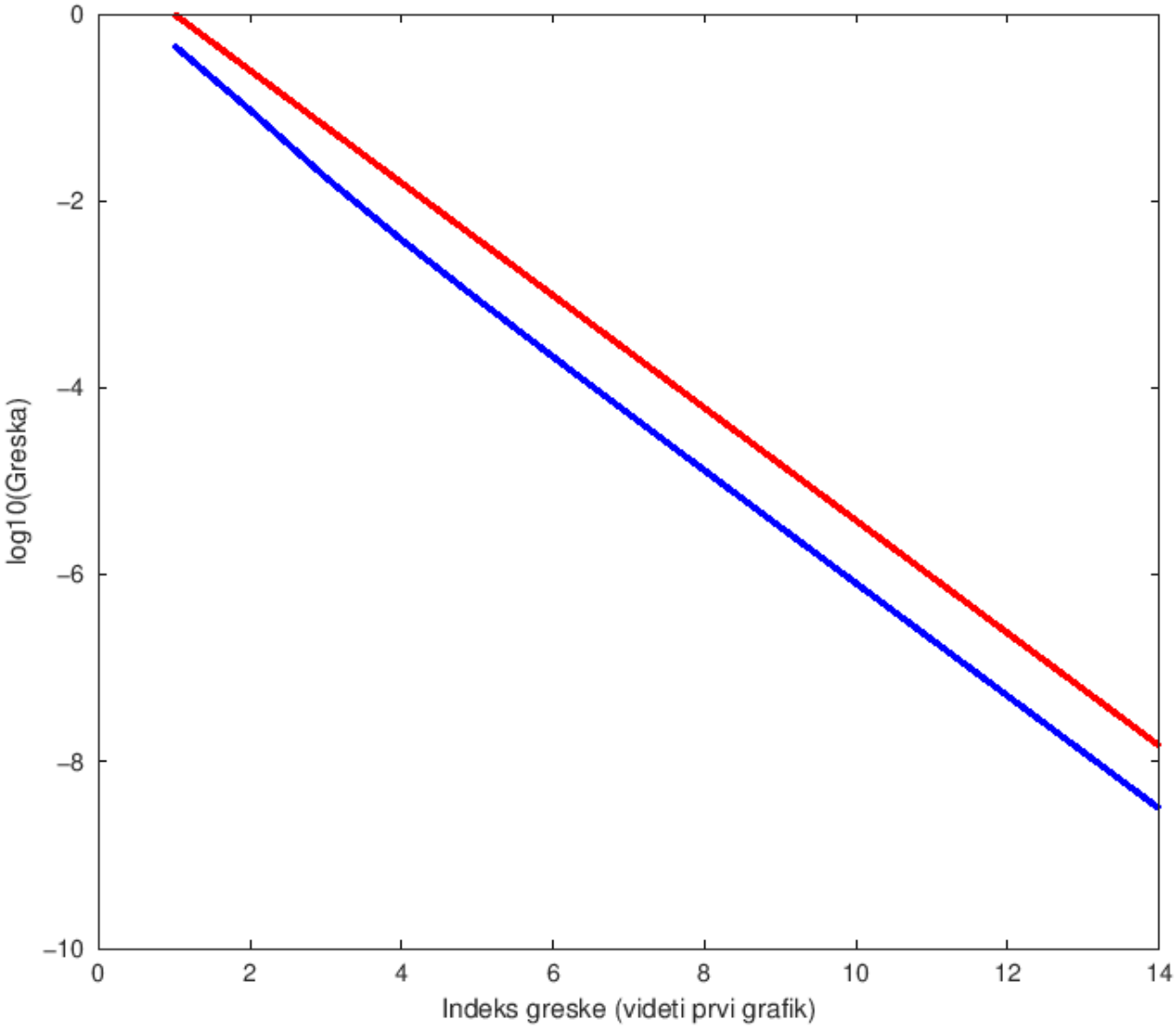


Poredimo sada grešku metoda srednje tačke (plavo) i funkcije $f(h) = h^2$ (crveno).

In [41]:
```
plot(1:length(errors),errors,"linewidth",10,"color","blue")
hold on;
plot(1:length(sub_intervals),sub_intervals.^2,"linewidth",10,"color", "red")
xlabel('Indeks greske (videti prvi grafik)');
ylabel('Greska');
set(gca, "fontsize", 14)
```

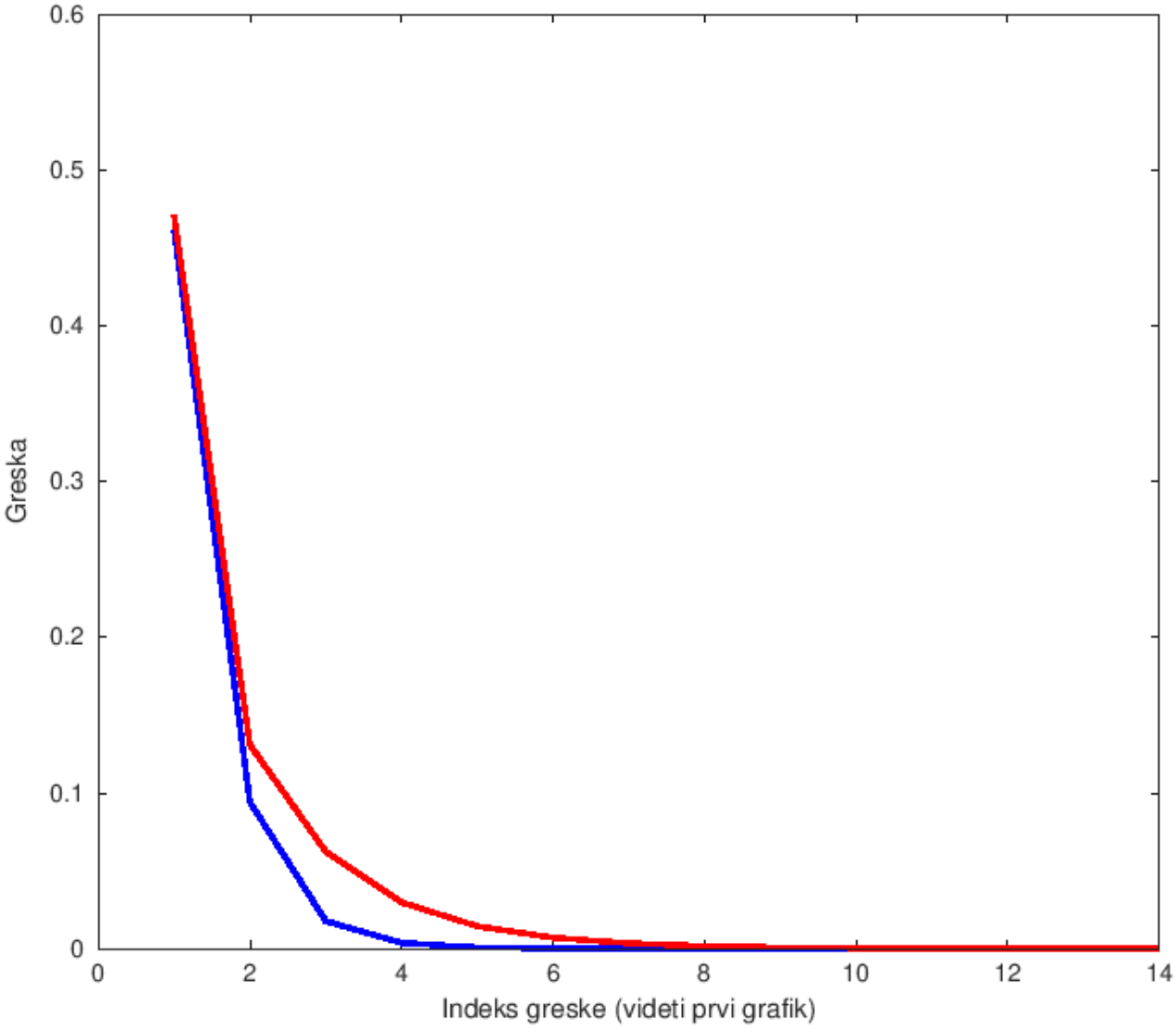


In [42]:
```
plot(1:length(errors),log10(errors),"linewidth",10,"color","blue")
hold on;
plot(1:length(sub_intervals),log10(sub_intervals.^2),"linewidth",10,"color", "red")
xlabel('Indeks greske (videti prvi grafik)');
ylabel('log10(Greska)');
set(gca, "fontsize", 14)
```

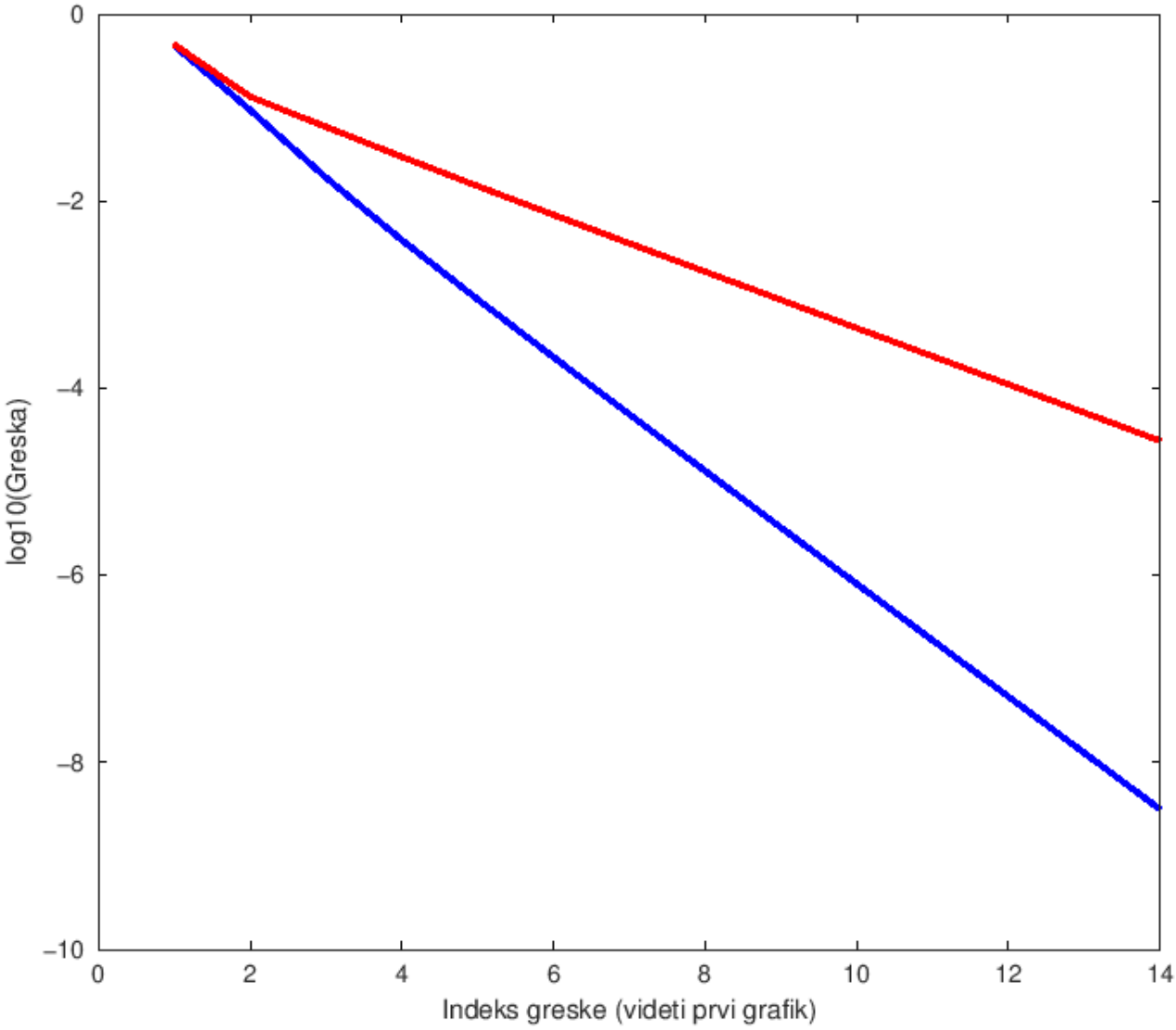


Poredimo sada grešku metoda srednje tačke (plavo) i Ojlerovog metoda (crveno).

In [44]:
```
plot(1:length(errors),errors,"linewidth",10,"color","blue")
hold on;
plot(1:length(errors),errors_ojler,"linewidth",10,"color", "red")
xlabel('Indeks greske (videti prvi grafik)');
ylabel('Greska');
set(gca, "fontsize", 14)
```



In [45]:
```
plot(1:length(errors),log10(errors),"linewidth",10,"color","blue")
hold on;
plot(1:length(errors),log10(errors_ojler),"linewidth",10,"color", "red")
xlabel('Indeks greske (videti prvi grafik)');
ylabel('log10(Greska)');
set(gca, "fontsize", 14)
```



Heunov metod

Intuitivno, ovaj metod koristi prosek nagiba tangente u početnoj tački x_0 i tangente u sledećoj tački $x_1 = x_0 + h$ da stigne do tačke y_1.

Formula za Heunov metod je:

$$y_{i+1}^0 = y_i + hf(x_i, y_i)$$

$$y_{i+1}^1 = y_i + h\frac{1}{2}(f(x_i, y_i) + f(x_{i+1}, y_{i+1}^0))$$

gde je $y(x)$ funkcija koja čija je diferencijalna jednačina data sa $f(x, y), y(x_0)$, a h proizvoljno odabrani korak.

Pre nego što primenimo Heunov metod na primer, objasnićemo ga pomoću grafika.

Na slici ispod dat je prvi korak Heunovog metoda.

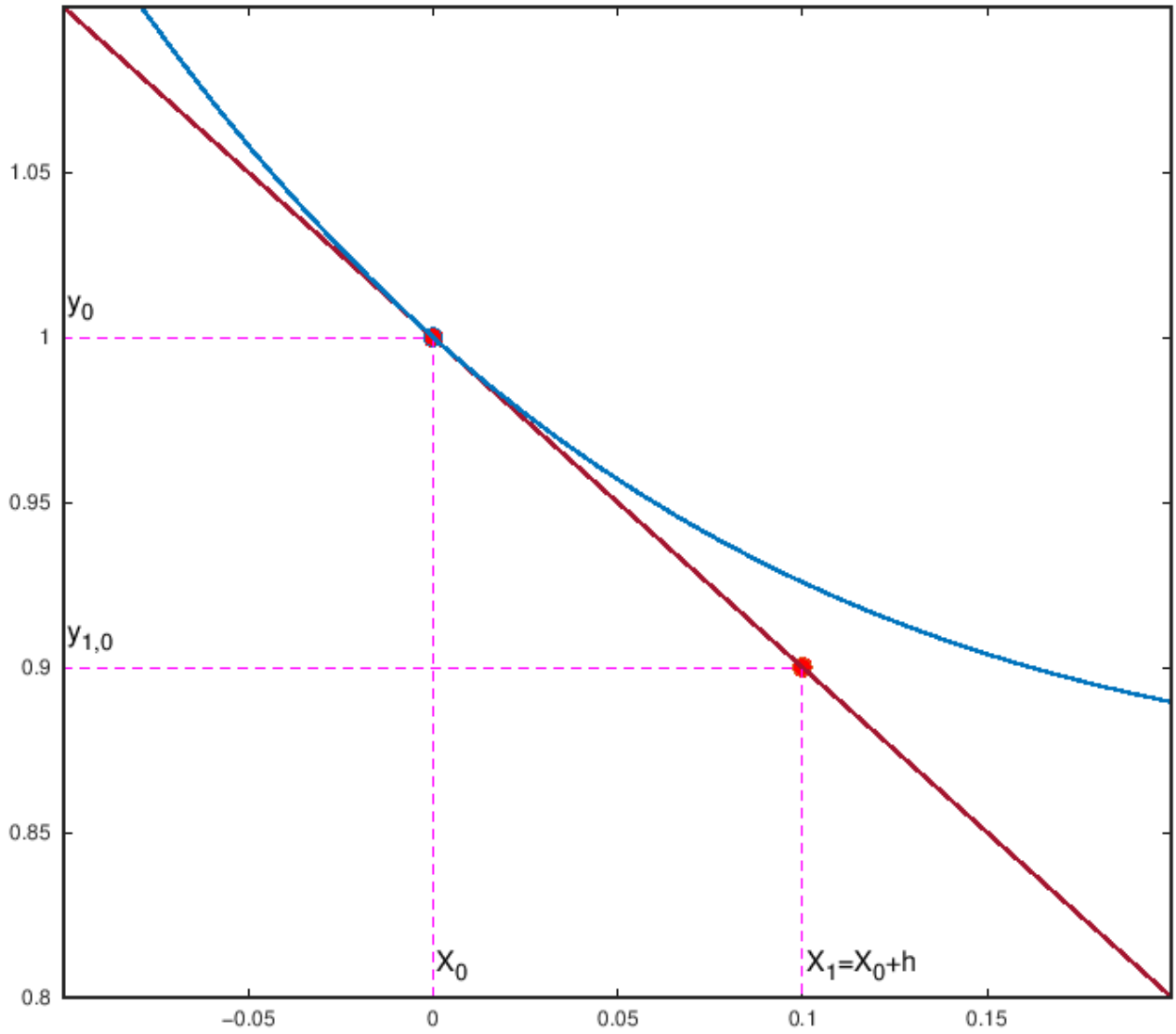
Primenjujemo sada Heunov na primer $f(x, y) = 2 - e^{-4x} - 2y,\ \ y(0) = 1$, za $h = 0.1$.

$$y_1^0 = y(0.1)^0 = y(0) + 0.1f(0, 1) = 1 + 0.1 \cdot (2 - e^{-4 \cdot 0} - 2 \cdot 1) = 0.90$$

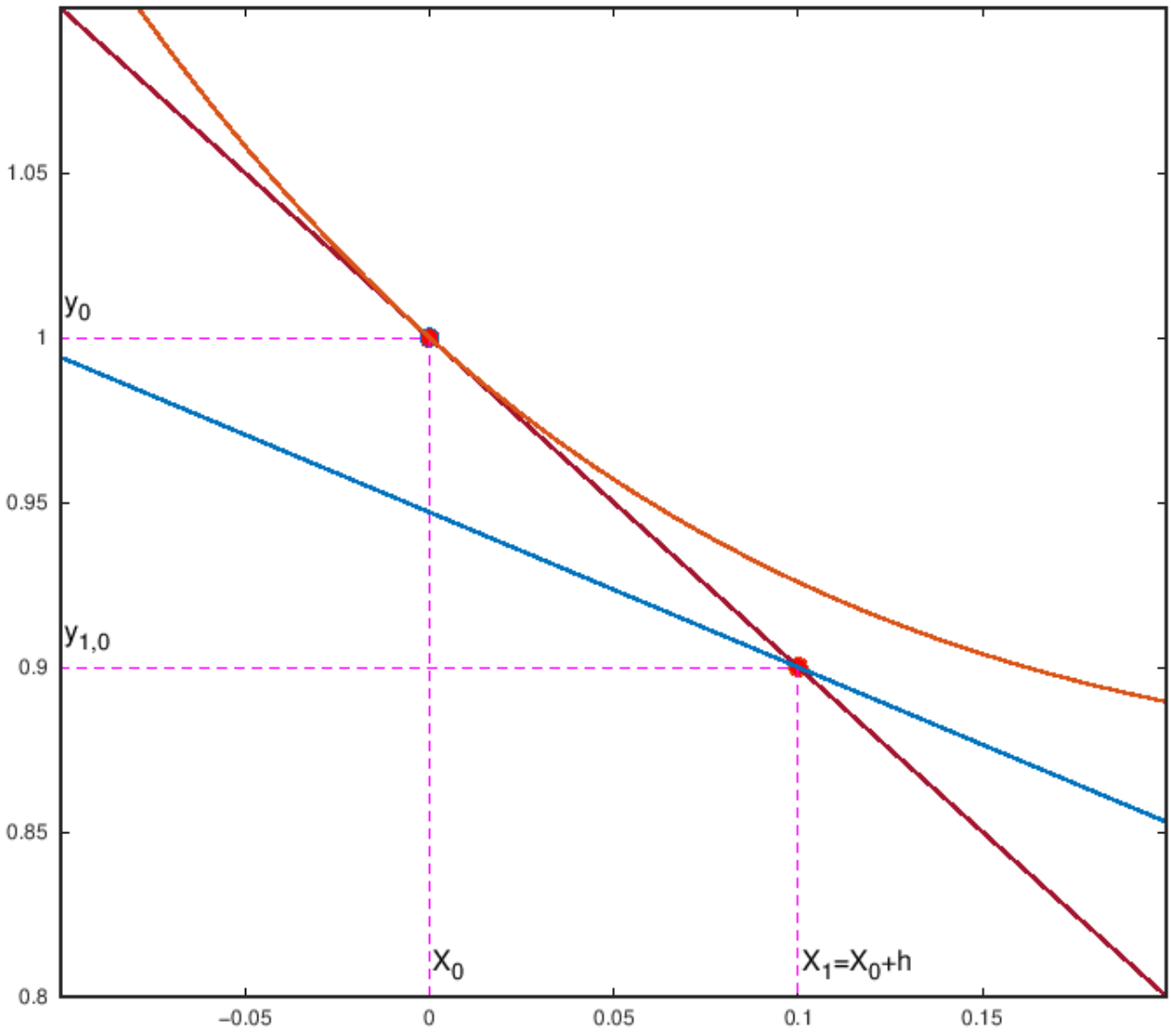
$$y_1 = y(0.1) = y(0) + \frac{0.1}{2}(f(0, 1) + f(0.1, 0.90)) = 1 + \frac{0.1}{2} \cdot (2 - e^{-4 \cdot 0} - 2 \cdot 1 + 2 - e^{-4 \cdot 0.1} - 2 \cdot 0.90) = 0$$

Prikazaćemo sada grafički jedan korak metoda srednje tačke baš za ovaj primer, a nakon toga pišemo kod za metod srednje tačke.

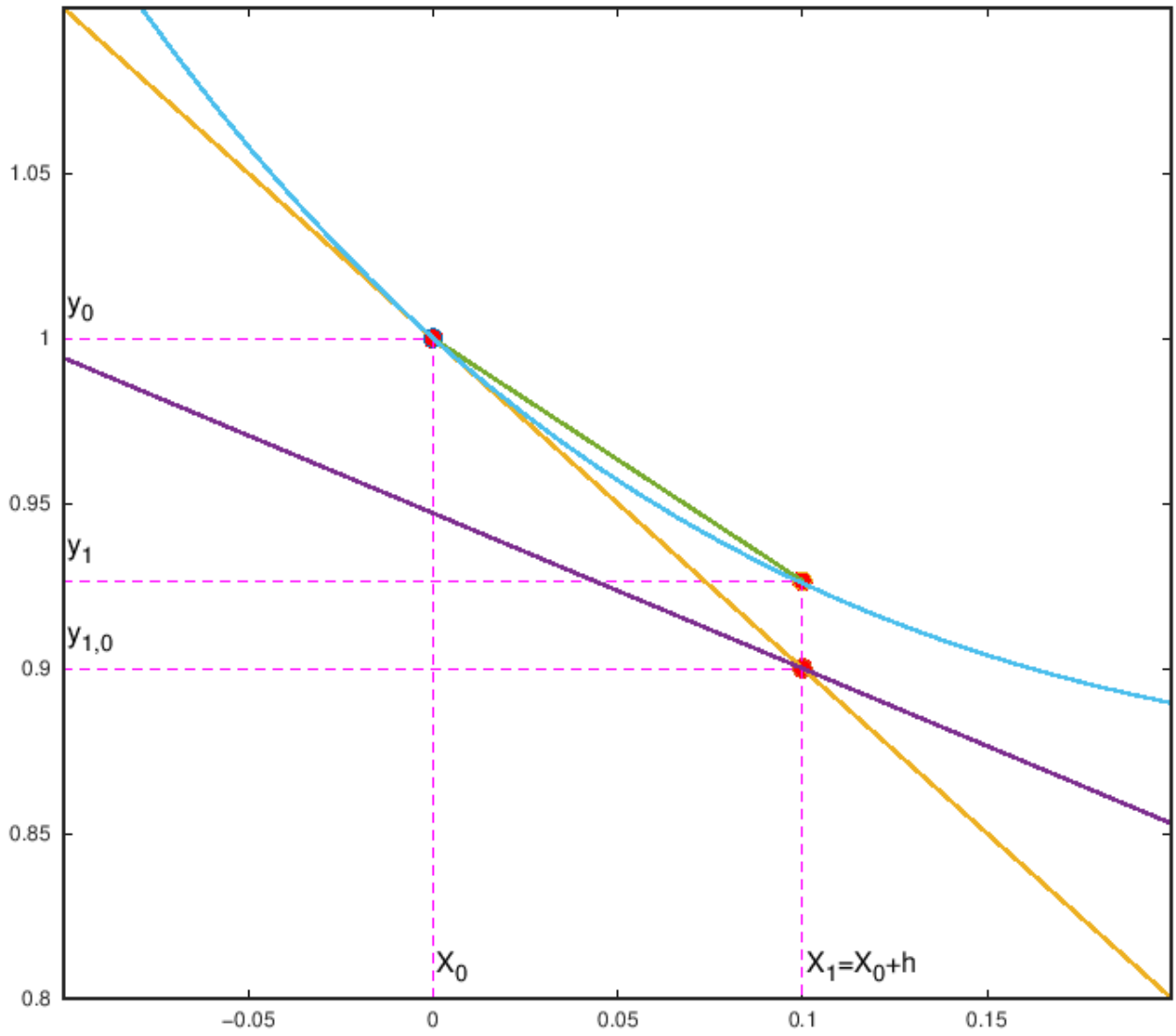
In [46]:
```
ode=@(x,y)2-exp(-4.*x)-2.*y;
fun=@(x,y)1+1/2.*exp(-4.*x)-1/2.*exp(-2.*x);
x0=0;y0=1;xn=3;h=0.1;
plot_he_st_1(x0,y0,h,ode)
plot_function([x0-0.1,xn],fun)
```



In [47]:
```
plot_he_st_2(x0,y0,h,ode)
plot_function([x0-0.1,xn],fun)
```



In [48]:
```
plot_he_st_3(x0,y0,h,ode)
plot_function([x0-0.1,xn],fun)
```



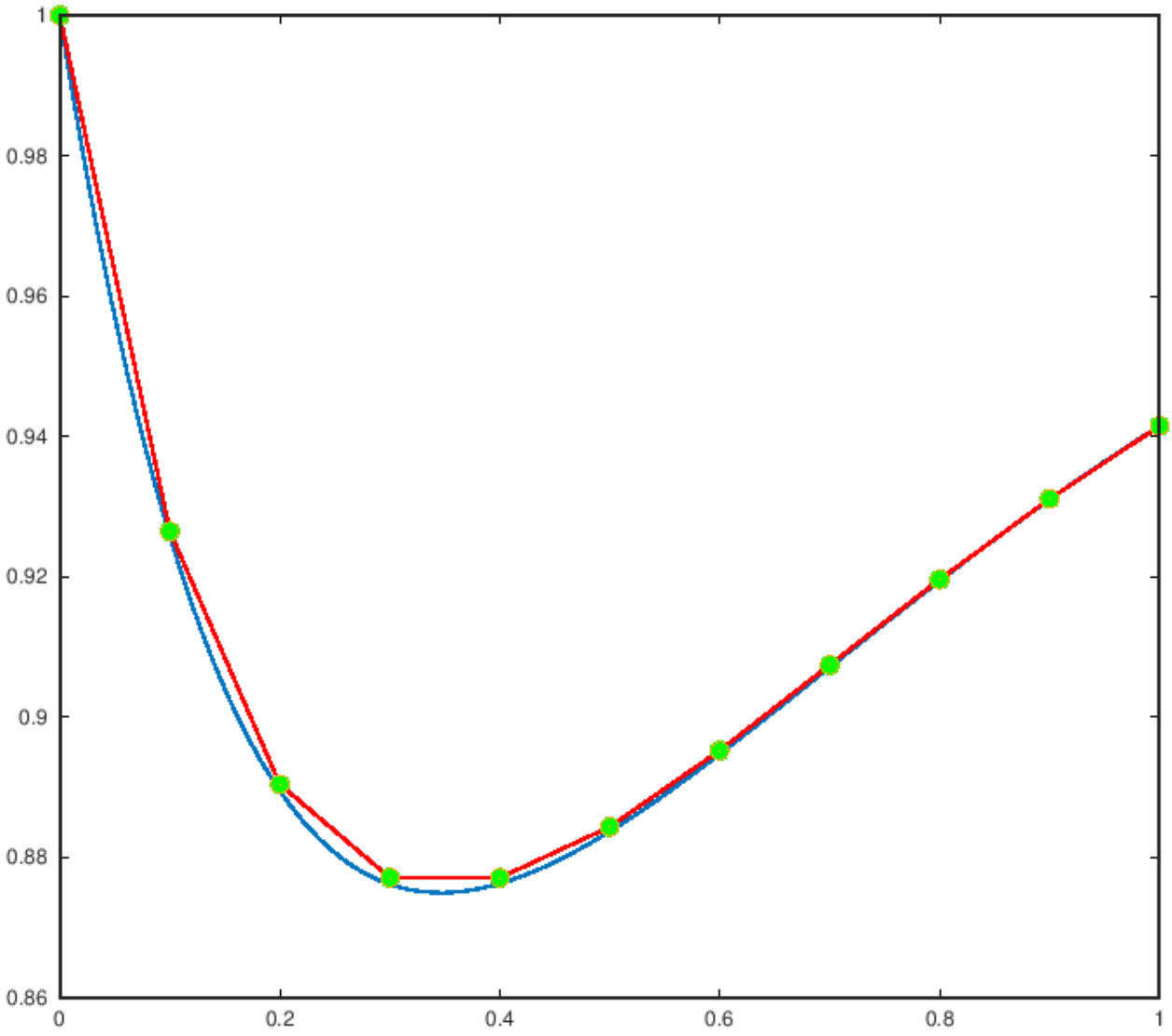
In [49]:
```
function y=heun(x0,xn,y0,h,odj)
    x = x0:h:xn;
    n = length(x);
    y = zeros(1,n);
    y(1) = y0;
    for i=2:n
        y(i) = y(i-1) + h*feval(odj, x(i-1), y(i-1));
        y(i) = y(i-1) + h*(feval(odj, x(i-1), y(i-1))+feval(odj, x(i), y(i)))/2;
    end
endfunction
```

In [51]:
```
ode=@(x,y)2-exp(-4.*x)-2.*y;
x0=0;y0=1;xn=1;h=0.1;
y=heun(x0,xn,y0,h,ode)
y_heun=y;
```

```
y =

 Columns 1 through 8:

   1.00000   0.92648   0.89044   0.87713   0.87710   0.88438   0.89524   0.90743

 Columns 9 through 11:

   0.91962   0.93109   0.94149
```

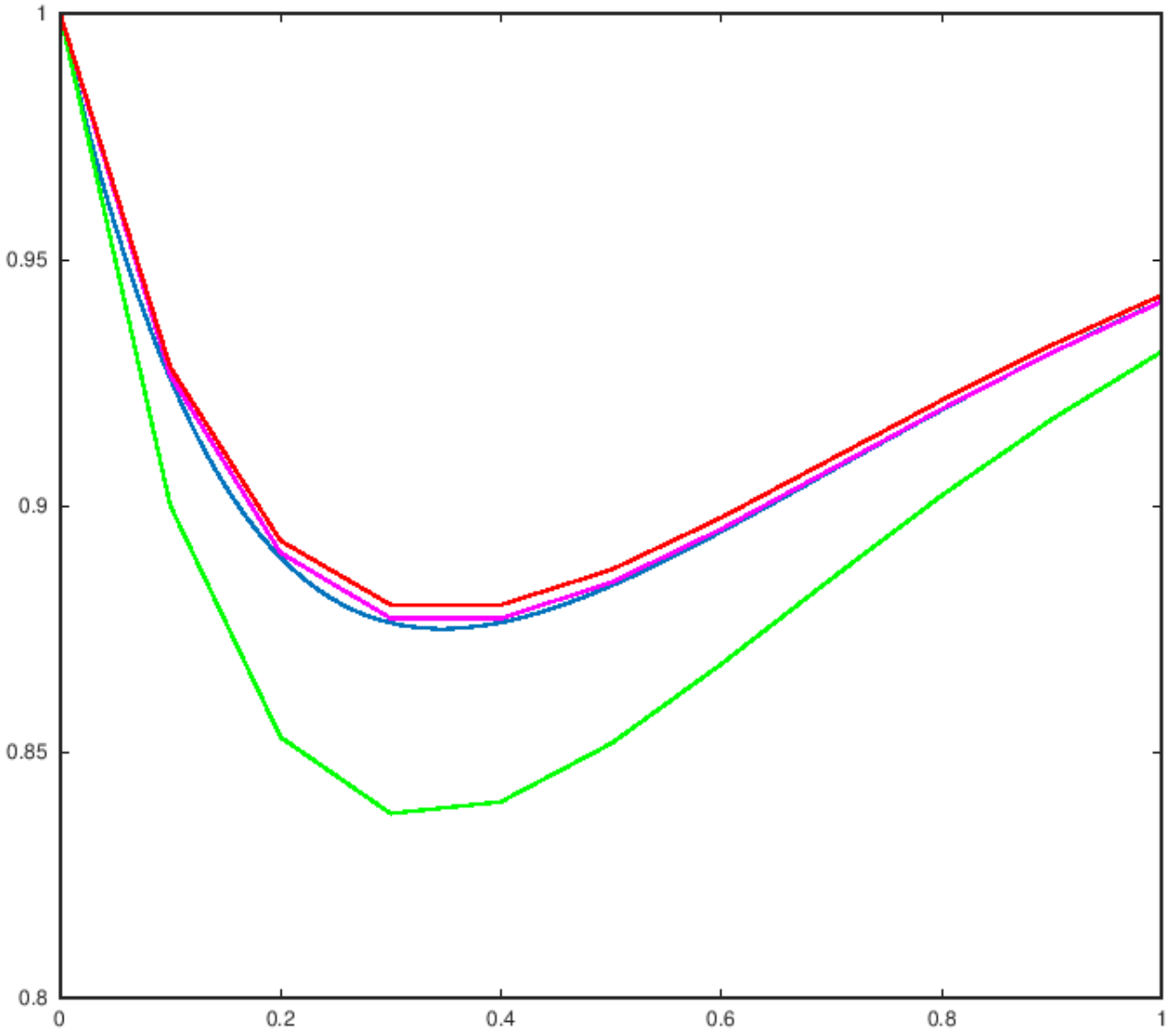
In [52]:
```
p=linterp(x0:h:xn,y)
plot_function([x0,xn],fun);
hold on;
plot(xp,polyval(p,xp),"linewidth", 5,"color", "red");
plot(x0:h:xn,y,'o','markersize', 10,'markerfacecolor','g');
```

```
p =

 Columns 1 through 7:

   0.020481  -0.154169   0.570051  -1.410858   2.660315  -4.032074   4.913713

 Columns 8 through 11:

  -4.598048   2.961990  -0.989914   1.000000
```



Poredimo Ojlerov metod (zeleno), metod srednje tačke (crveno) i Henov metod (magenta).

In [53]:
```
plot_function([x0,xn],fun);
hold on;
p_ojler=linterp(x0:h:xn,y_ojler);
p_stacka=linterp(x0:h:xn,y_stacka);
plot(xp,polyval(p,xp),"linewidth", 5,"color", "magenta");
plot(xp,polyval(p_ojler,xp),"linewidth", 5,"color", "green");
plot(xp,polyval(p_stacka,xp),"linewidth", 5,"color", "red");
```

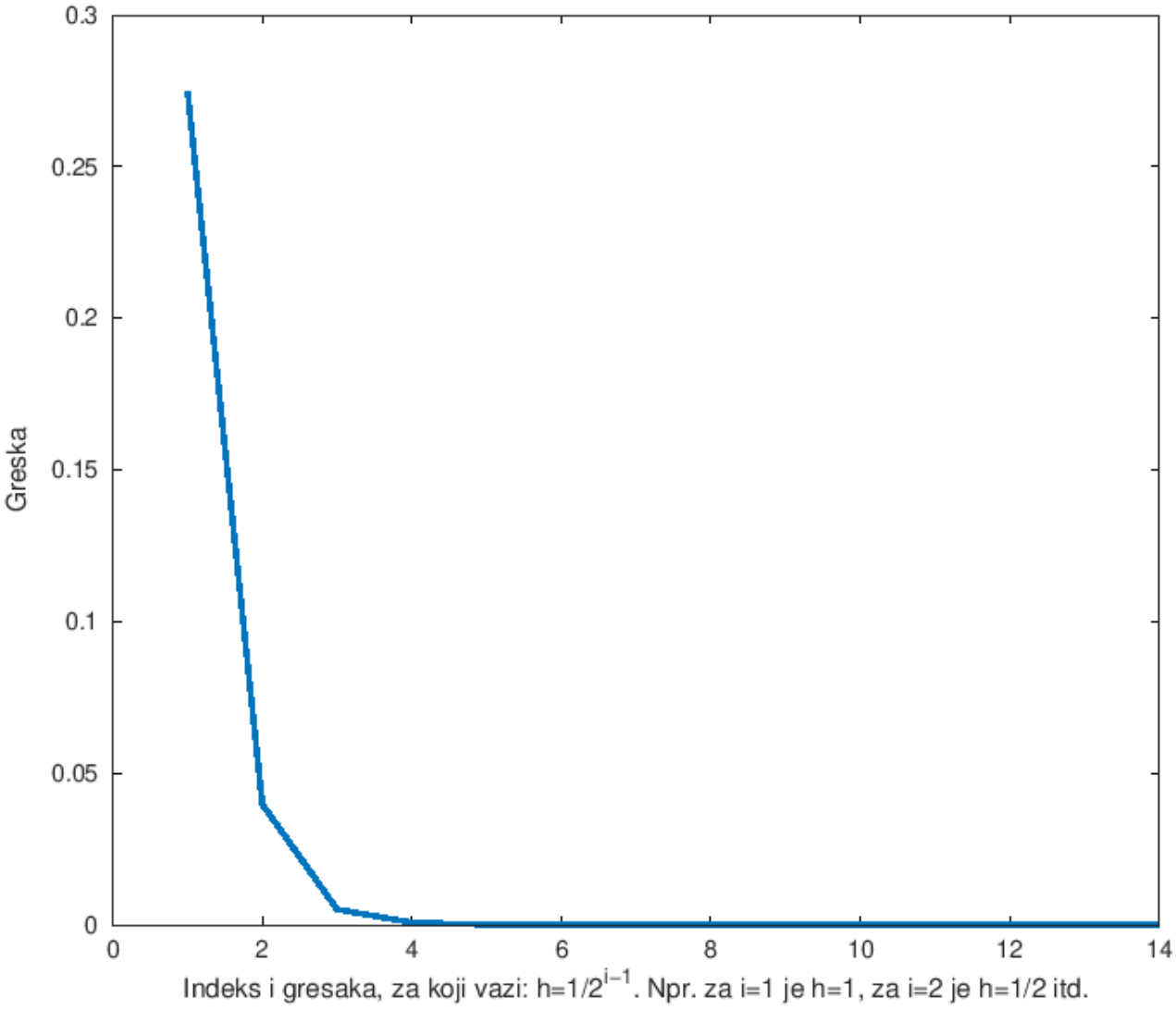


Vidimo da Heunov metod ima sličnu tačnost kao metod srednje tačke. Formalno tačnost Heunovog metoda je:

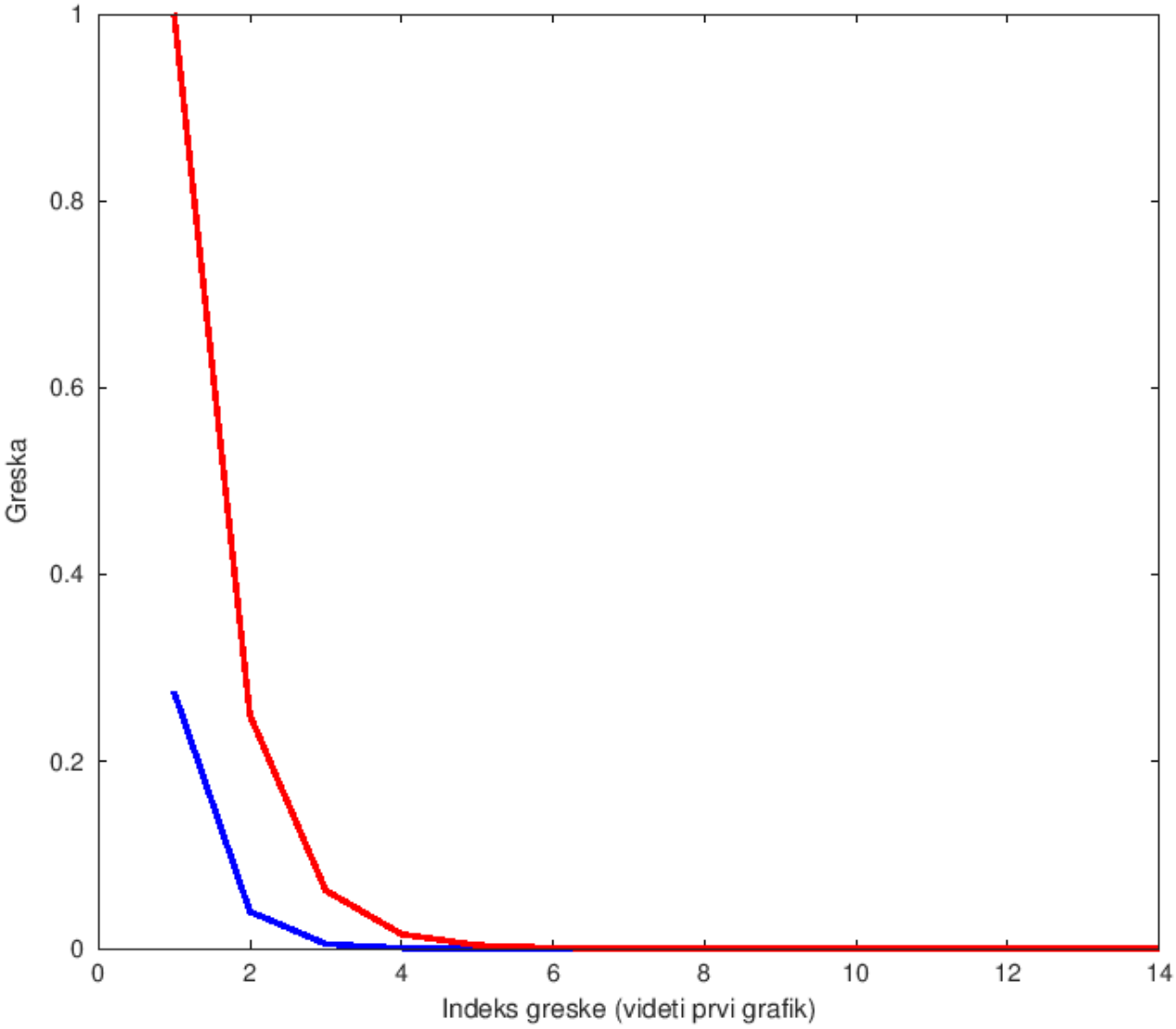
$$O(h^2)$$

Kasnije objašnjavamo zato red greške ima baš ovu vrednost. Pokazaćemo sada grafike greške za Heunov metod.

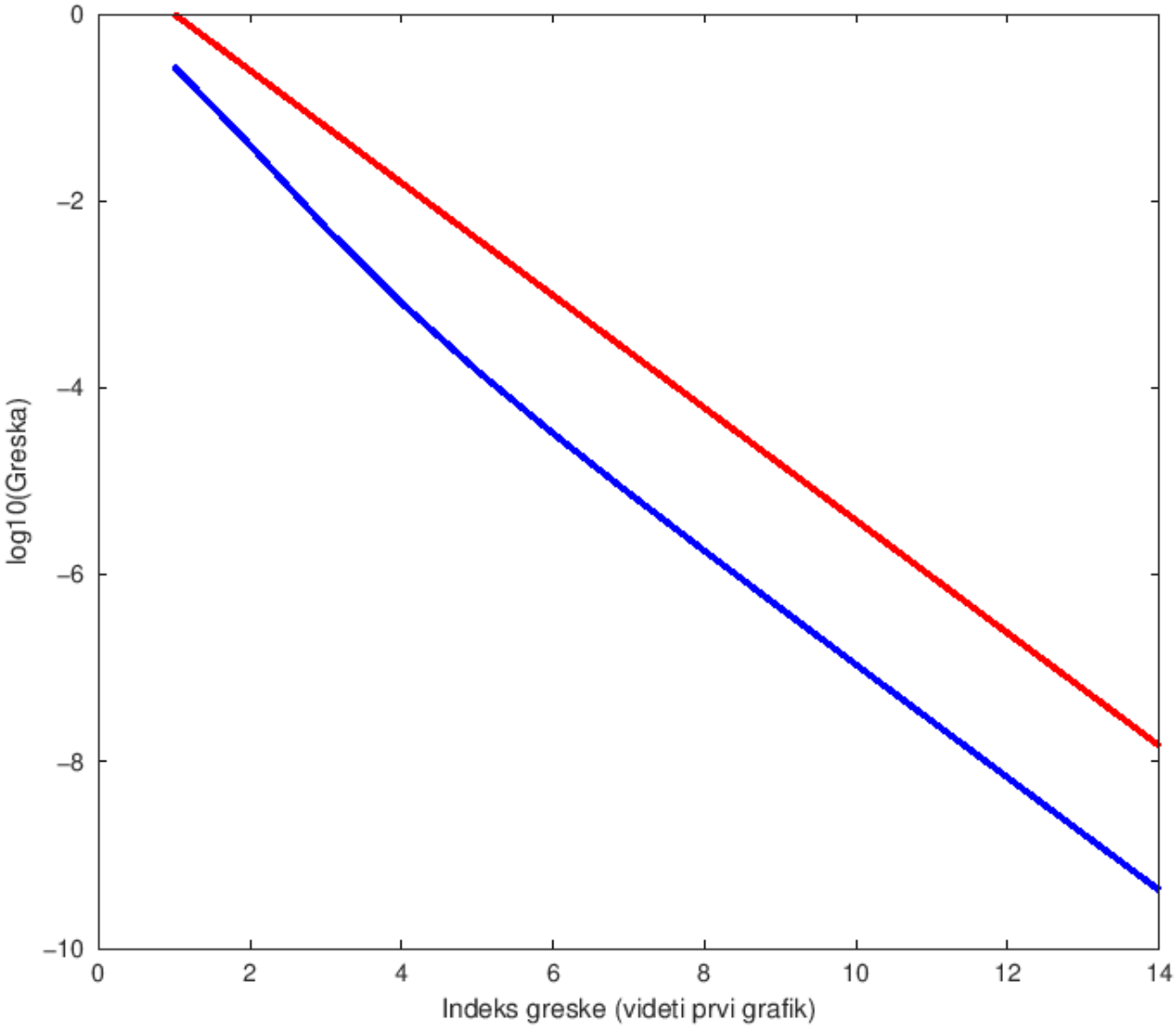
In [54]:
```
[errors,sub_intervals]=calculate_error(x0,xn,y0,ode,fun,0.0001,'heun');
errors_heun=errors;
```



In [55]:
```
plot(1:length(errors),errors,"linewidth",10,"color","blue")
hold on;
plot(1:length(sub_intervals),sub_intervals.^2,"linewidth",10,"color", "red")
xlabel('Indeks greske (videti prvi grafik)');
ylabel('Greska');
set(gca, "fontsize", 14)
```

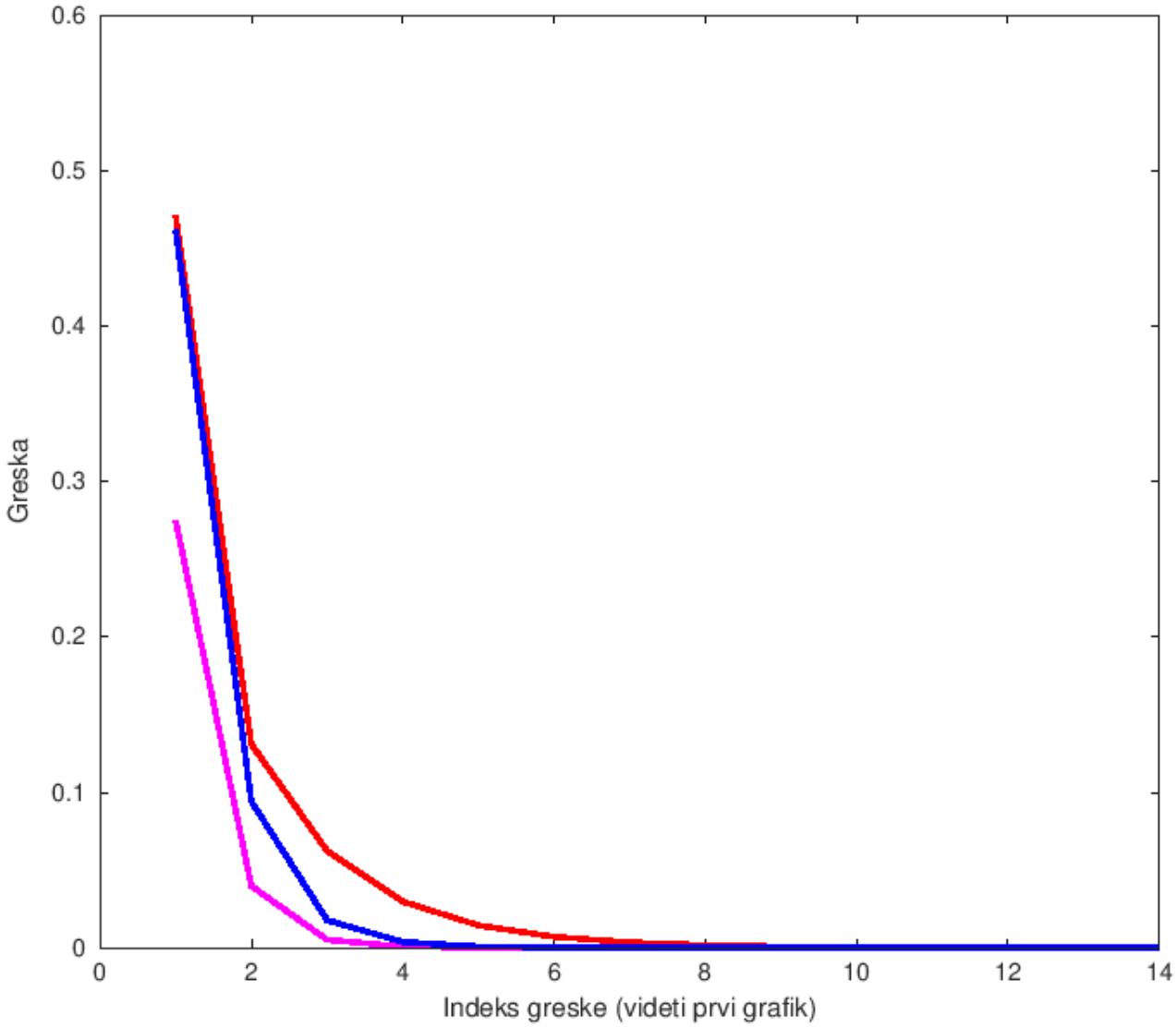


In [56]:
```
plot(1:length(errors),log10(errors),"linewidth",10,"color","blue")
hold on;
plot(1:length(sub_intervals),log10(sub_intervals.^2),"linewidth",10,"color", "red")
xlabel('Indeks greske (videti prvi grafik)');
ylabel('log10(Greska)');
set(gca, "fontsize", 14)
```

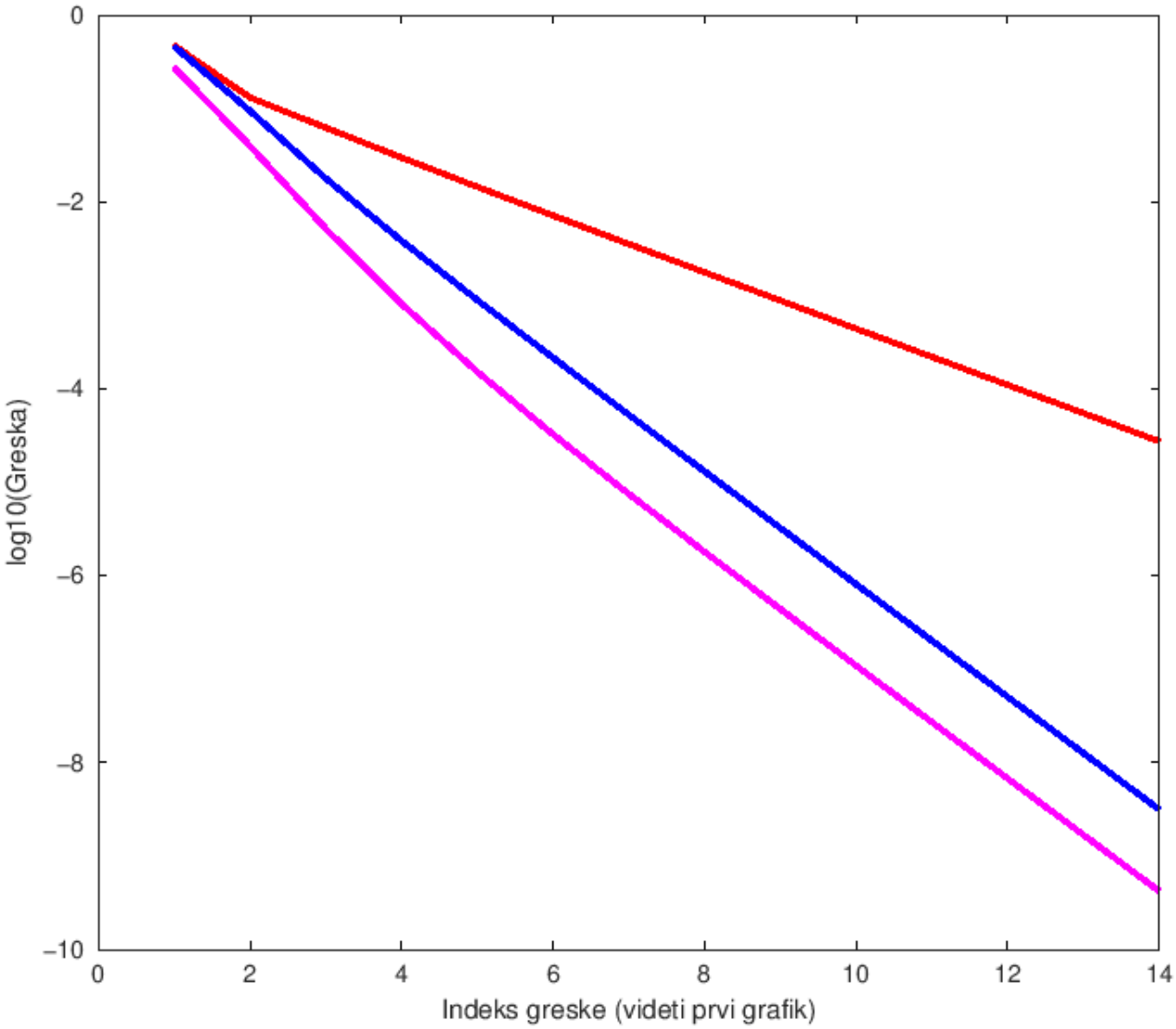


Poredimo sada grešku Ojlerovog metoda (crveno), metoda srednje tačke (plavo) i Heunovog metoda (magenta).

In [57]:
```
plot(1:length(errors),errors,"linewidth",10,"color","magenta")
hold on;
plot(1:length(errors),errors_ojler,"linewidth",10,"color", "red")
plot(1:length(errors),errors_stacka,"linewidth",10,"color","blue")
xlabel('Indeks greske (videti prvi grafik)');
ylabel('Greska');
set(gca, "fontsize", 14)
```



In [58]:
```
plot(1:length(errors),log10(errors),"linewidth",10,"color","magenta")
hold on;
plot(1:length(errors),log10(errors_ojler),"linewidth",10,"color", "red")
plot(1:length(errors),log10(errors_stacka),"linewidth",10,"color","blue")
xlabel('Indeks greske (videti prvi grafik)');
ylabel('log10(Greska)');
set(gca, "fontsize", 14)
```



Runge-Kutta metode

Kao što smo iz Heunovog i metoda srednje tačke videli moguće je dobiti tačnost većeg reda od $O(h)$ bez korišćenja drugog izvoda. U nastavku ćemo objasniti na koji način su izvedena ta dva metoda.

Pretpostavka od koje polaze Runge-Kutta metode je da se prelazak iz tačke (x_0, y_0) u tačku (x_1, y_1) može kompletno parametrizovati. To znači da je dozvoljeno biranje bilo kojih tangenti u bilo kojim tačkama da bi se prelaz uradio. Postavlja se pitanje sa kojim ciljem (ograničenjima) biramo tangente.

Cilj je da postignemo veću tačnost odnosno isti rezultat kao kada bi imali drugi, treći, četvrti itd. izvod.

Ideja je vrlo slična Gausovoj kvadraturi.

Kao što smo kod Gausove kvadrature birali broj tačaka u ovom slučaju prvo biramo koliko ćemo tangenti koristiti pa onda izvodimo metod. Po tome se razlikuje red Runge-Kutta metoda. Detaljno ćemo objasniti Runge-Kutta metode drugog reda.

Runge-Kutta metode drugog reda

Cilj nam je da koristimo dve tangente i da postignemo tačnost kao da koristimo Tejlorov red sa tri člana, tj. kao da imamo drugi izvod.

Zapisujemo prvo parametrizovani oblik prelaza od (x_0, y_0) u tačku (x_1, y_1):

$$y_{i+1} = y_i + w_1 K_1 + w_2 K_2$$

$$K_1 = hf(x_i, y_i)$$

$$K_2 = hf(x_i + \alpha h, y_i + \beta K_1)$$

, gde su α, β, w_1, w_2 parametri, tj. nepoznati koeficijenti koje treba da odredimo tako da dobijemo tačnost Tejlorovog reda sa tri člana.

Dajemo prvo formulu za Tejlorov red sa tri člana:

$$y_{i+1} = y_i + hf(x_i, y_i) + \frac{h}{2} f'(x_i, y_i) + O(h^3)$$

Napomenućemo još jednom da nemamo drugi izvod, tj. nemamo prvi izvod diferencijalne jednačine $f'(x_i, y_i)$.

Sredićemo sada malo formulu za Tejlorov red tako što ćemo razviti $f'(x_i, y_i)$.

$$f'(x, y) = \frac{\delta f(x, y)}{\delta x} + \frac{\delta f(x, y)}{\delta y} \frac{dy}{dx} = \frac{\delta f(x, y)}{\delta x} + \frac{\delta f(x, y)}{\delta y} f(x, y)$$

$$y_{i+1} = y_i + hf(x_i, y_i) + \frac{h}{2}\frac{\delta f(x,y)}{\delta x} + \frac{h}{2}\frac{\delta f(x,y)}{\delta y} f(x,y)$$

Ponavljamo sada parametrizovani oblik u jednom redu:

$$y_{i+1} = y_i + w_1 K_1 + w_2 K_2 = w_1 hf(x_i, y_i) + w_2 hf(x_i + \alpha h, y_i + \beta K_1)$$

Da bi prethodnu formulu malo sredili upotrebićemo Tejlorov red za funkciju sa dve promenljive:

$$f(x_0 + h, y_0 + k) = f(x_0, y_0+) + \frac{\delta f(x_0, y_0)}{\delta x} h + \frac{\delta f(x_0, y_0)}{\delta y} k + \dots.$$

$$f(x_i + \alpha, y_i + \beta) = f(x_i, y_i) + \frac{\delta f(x_i, y_i)}{\delta x} \alpha + \frac{\delta f(x_i, y_i)}{\delta y} \beta + \dots.$$

$$y_{i+1} = y_i + w_1 hf(x_i, y_i) + w_2 hf(x_i + \alpha h, y_i + \beta K_1) = y_i + w_1 hf(x_i, y_i) + w_2 h(f(x_i, y_i) + \frac{\delta f(x_i, y_i)}{\delta x} \alpha + \frac{\delta f(x_i, y_i)}{\delta y} \beta)$$

$$y_{i+1} = y_i + (w_1 + w_2) hf(x_i, y_i) + w_2 h \frac{\delta f(x_i, y_i)}{\delta x} \alpha + w_2 h \frac{\delta f(x_i, y_i)}{\delta y} \beta$$

Izjednačavamo sada paremetrizovani oblik i Tejlorov red da bi videli koje uslove treba da zadovoljavaju parametri (slično kao kod Gausove kvadrature).

$$y_{i+1} = y_i + (w_1 + w_2) hf(x_i, y_i) + w_2 \alpha h \frac{\delta f(x_i, y_i)}{\delta x} + w_2 h \beta \frac{\delta f(x_i, y_i)}{\delta y}$$

$$y_{i+1} = y_i + hf(x_i, y_i) + \frac{h}{2}\frac{\delta f(x,y)}{\delta x} + \frac{h}{2}\frac{\delta f(x,y)}{\delta y} f(x,y)$$

Dobijamo sledeće uslove:

$$w_1 + w_2 = 1, \;\; w_2 \alpha = \frac{1}{2}, \;\; w_2 \beta = \frac{1}{2}$$

Pošto imamo 3 uslova, a 4 parametra, imamo neodređen sistem. To znači da imamo beskonačno mnogo Runge-Kutta metoda drugog reda. Svaki izbor w_1, w_2, α, β koji zadovoljava prethodne uslove je validan Runge-Kutta metod drugog reda i ima globalnu tačnost $O(h^2)$.

Recimo da odabremo sada sledeće vrednosti parametra:

$$\alpha = \beta = 1, w_1 = w_2 = \frac{1}{2}$$

$$K_1 = hf(x_i, y_i)$$

$$K_2 = hf(x_i + \alpha h, y_i + \beta K_1) = hf(x_i + h, y_i + K_1)$$

$$y_{i+1} = y_i + w_1 K_1 + w_2 K_2 = y_i + \frac{1}{2}(K_1 + K_2) = y_i + \frac{1}{2}h(f(x_i, y_i) + f(x_i + h, y_i$$

Dobili smo Heunov metod. Dakle, vidimo da je on Runge-Kutta metod drugog reda. Na sličan način, ako odberemo $\alpha = \beta = \frac{1}{2}, w_1 = 0 = w_2 = 1$ dobijamo metod srednje tačke.

Runge Kutta metod četvrtog reda

U nastavku ćemo pokazati jedan od Runge Kutta metoda četvrtog reda.

Ovaj metod se najčešće koristi u praksi i označava se sa RK4. Koristi kombinaciju četiri tangente i ima tačnost $O(h^4)$.

Daćemo prvo formulu, pa nakon toga grafički prikaz i kod.

$$k_1 = f(x_i, y_i)$$

$$k_2 = f(x_i + \frac{h}{2}, y_i + \frac{1}{2}k_1 h)$$

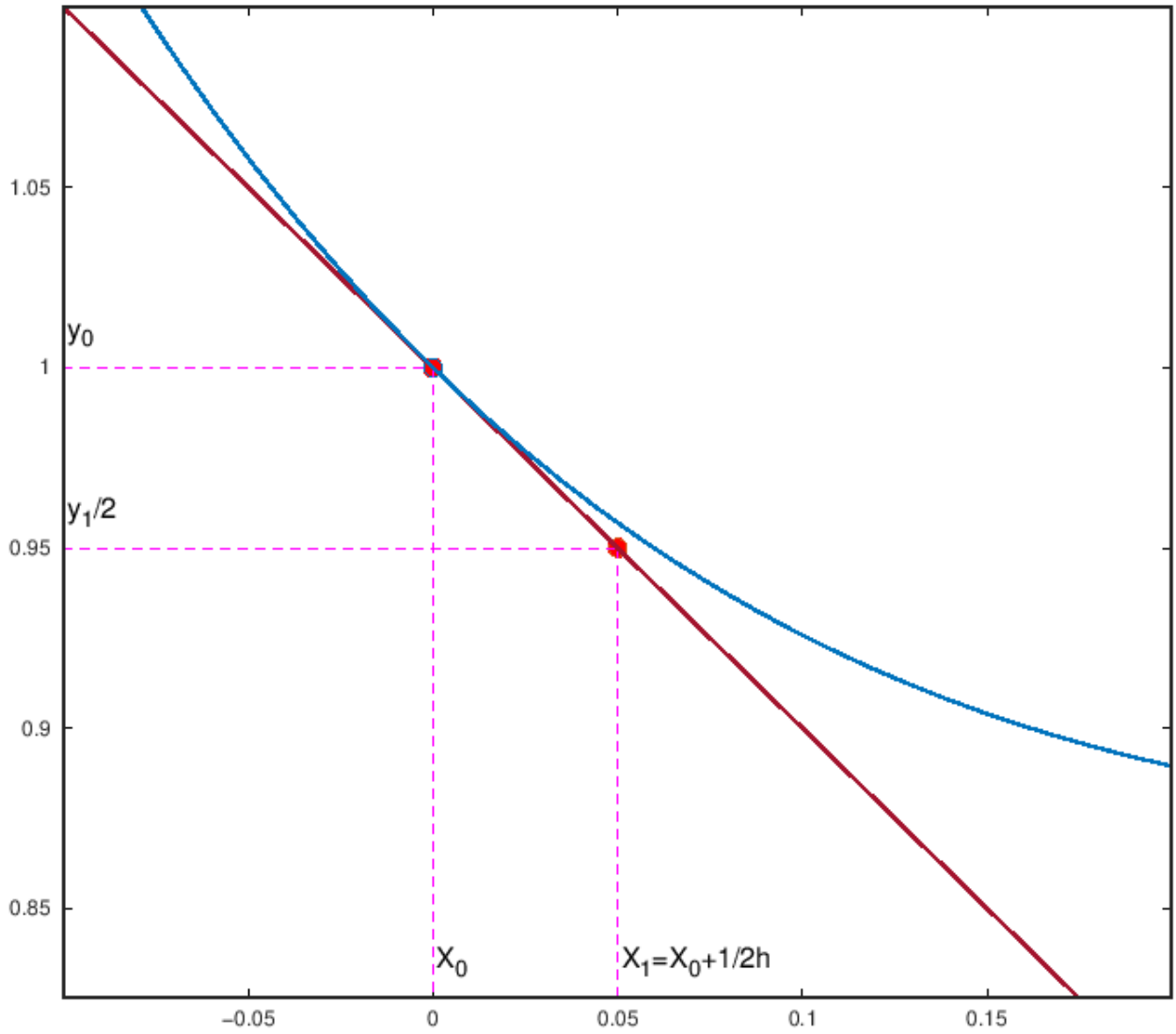
$$k_3 = f(x_i + \frac{h}{2}, y_i + \frac{1}{2}k_2 h)$$

$$k_4 = f(x_i + h, y_i + k_3 h)$$

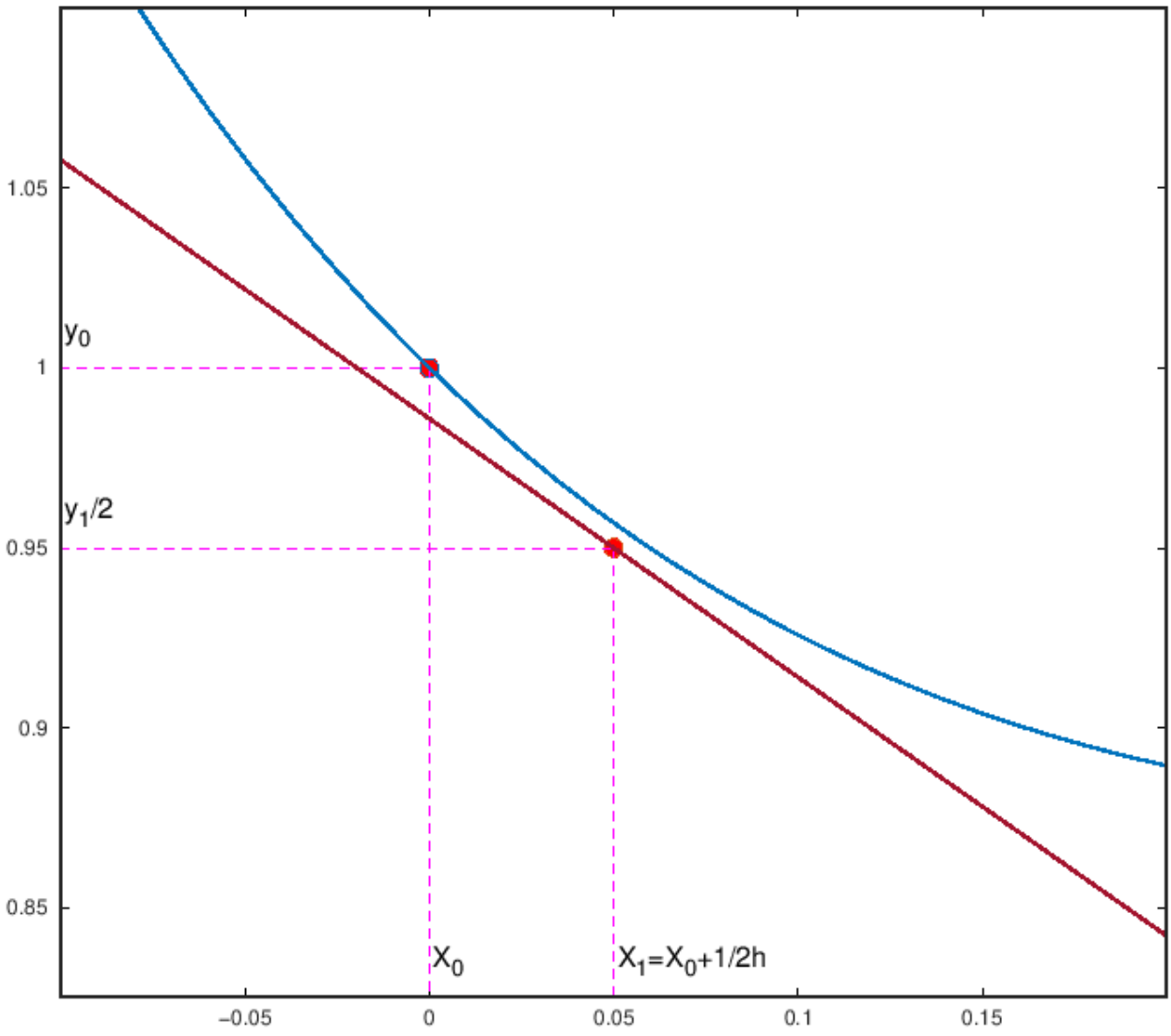
$$y_{i+1} = y_i + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

Prikazaćemo sada svaki korak (tangentu) koju RK4 metod koristi.

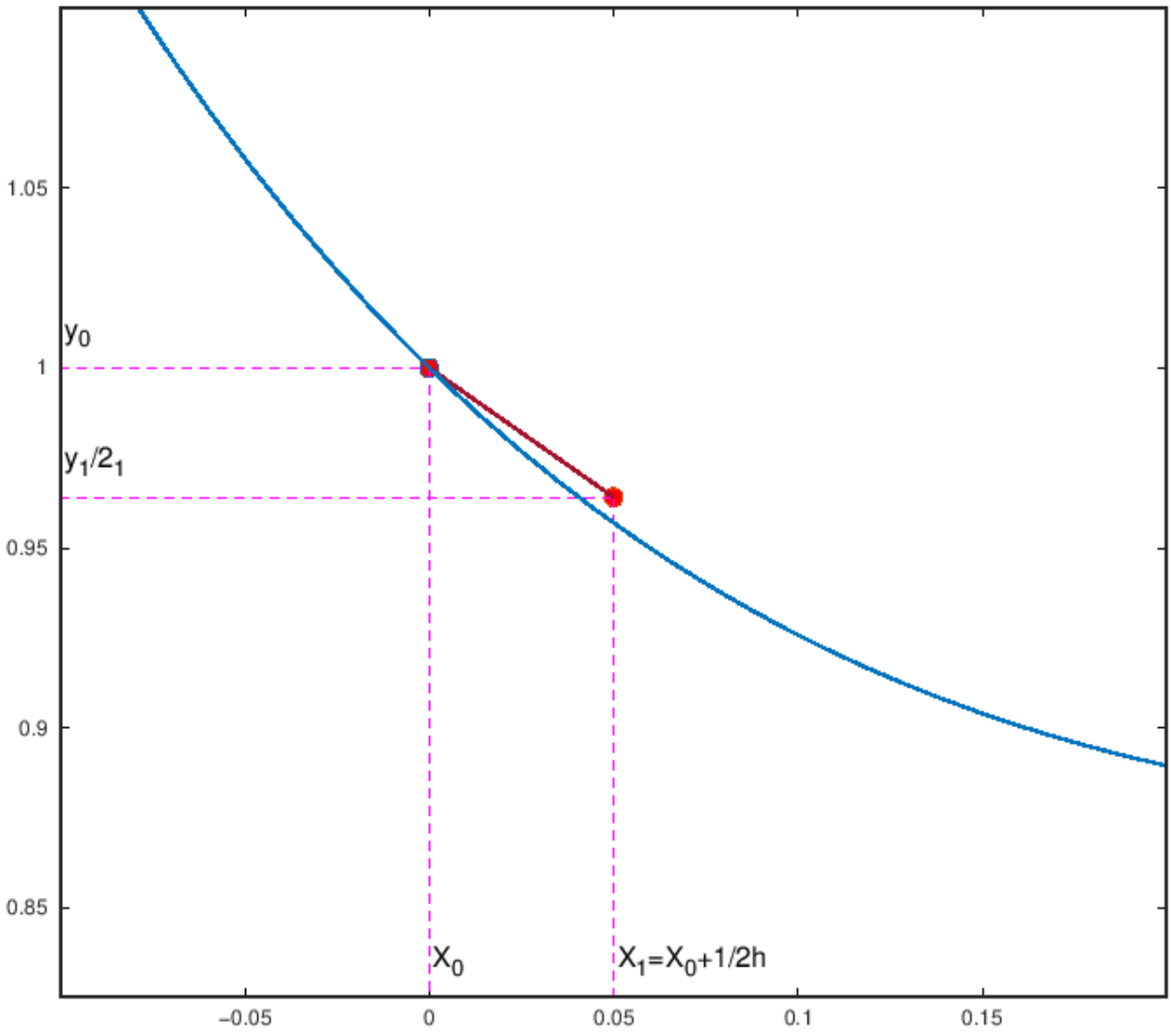
In [59]:
```
ode=@(x,y)2-exp(-4.*x)-2.*y;
fun=@(x,y)1+1/2.*exp(-4.*x)-1/2.*exp(-2.*x);
x0=0;y0=1;xn=3;h=0.1;
plot_rk4_st_1(x0,y0,h,ode)
plot_function([x0-0.1,xn],fun)
```



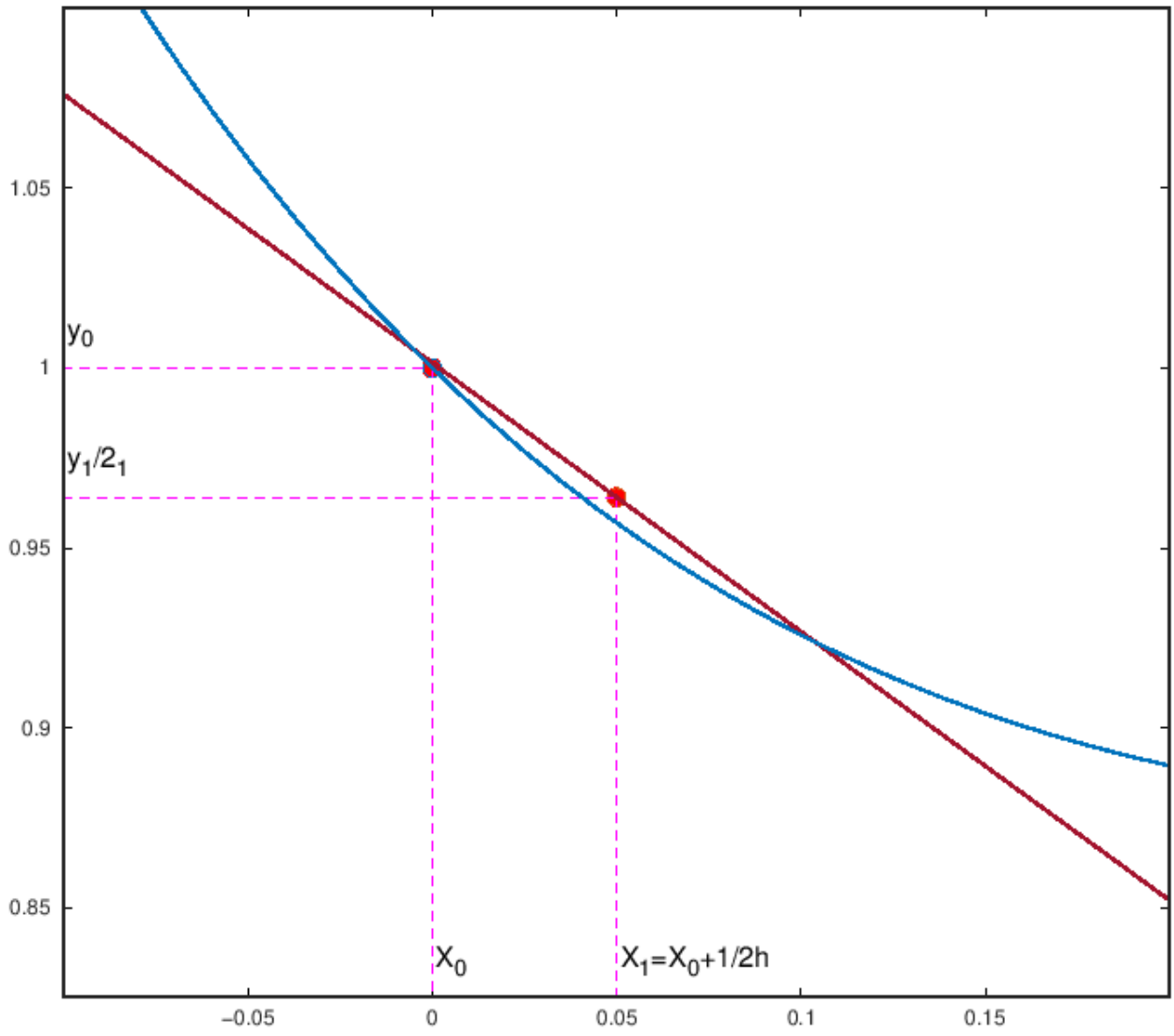
In [60]:
```
plot_rk4_st_2(x0,y0,h,ode)
plot_function([x0-0.1,xn],fun)
```



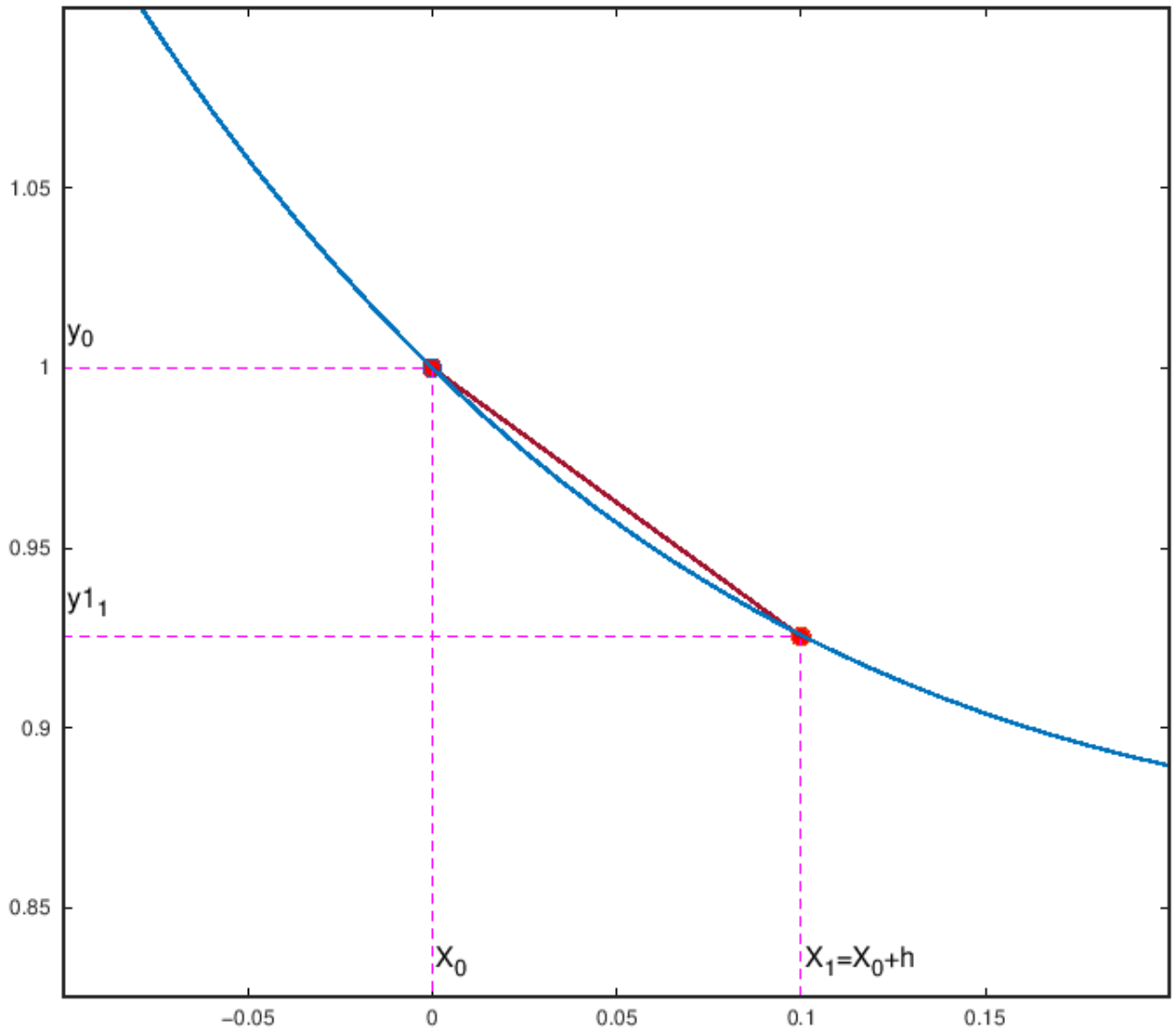
In [61]:
```
plot_rk4_st_3(x0,y0,h,ode)
plot_function([x0-0.1,xn],fun)
```



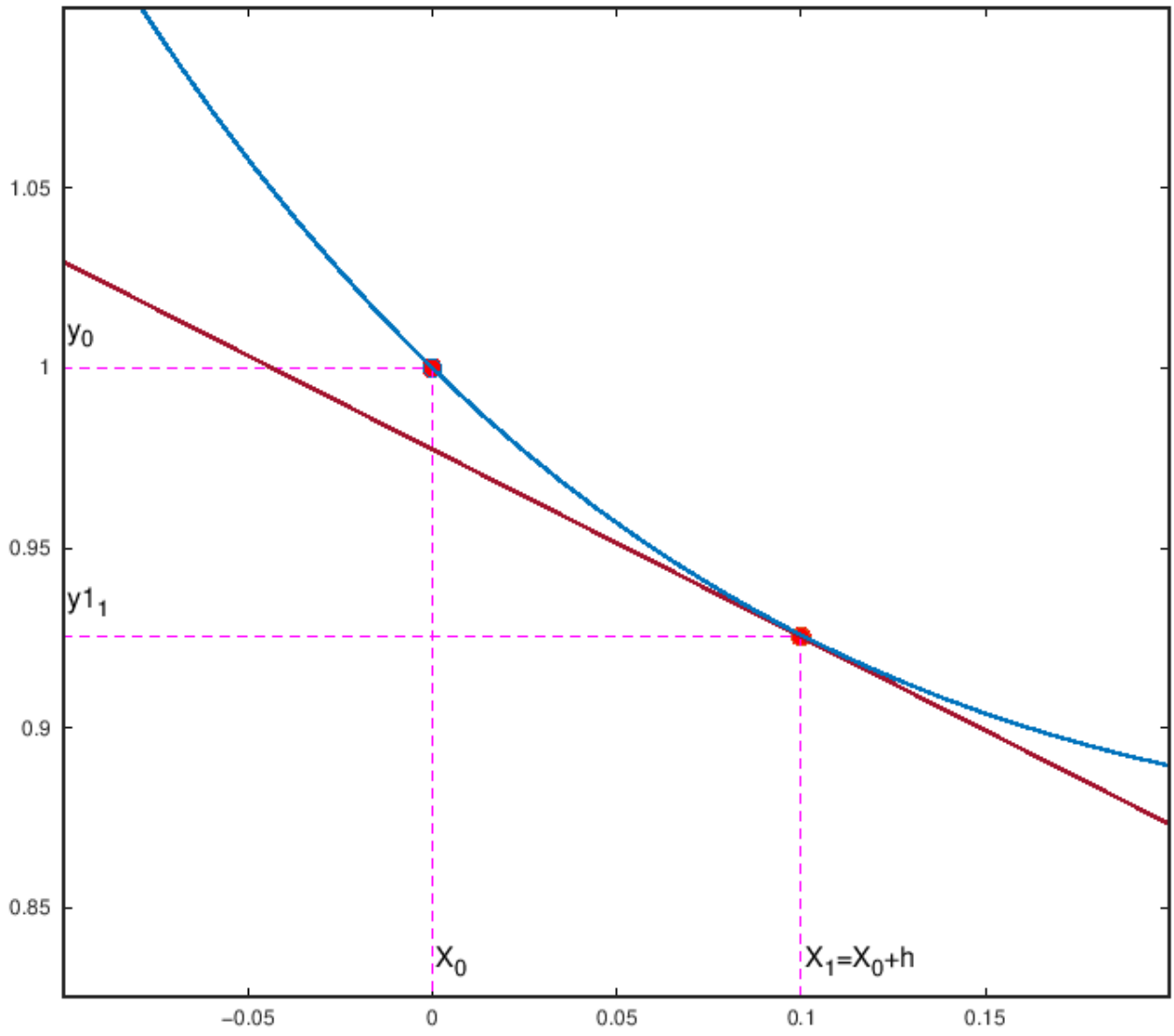
In [63]:
```
plot_rk4_st_4(x0,y0,h,ode)
plot_function([x0-0.1,xn],fun)
```



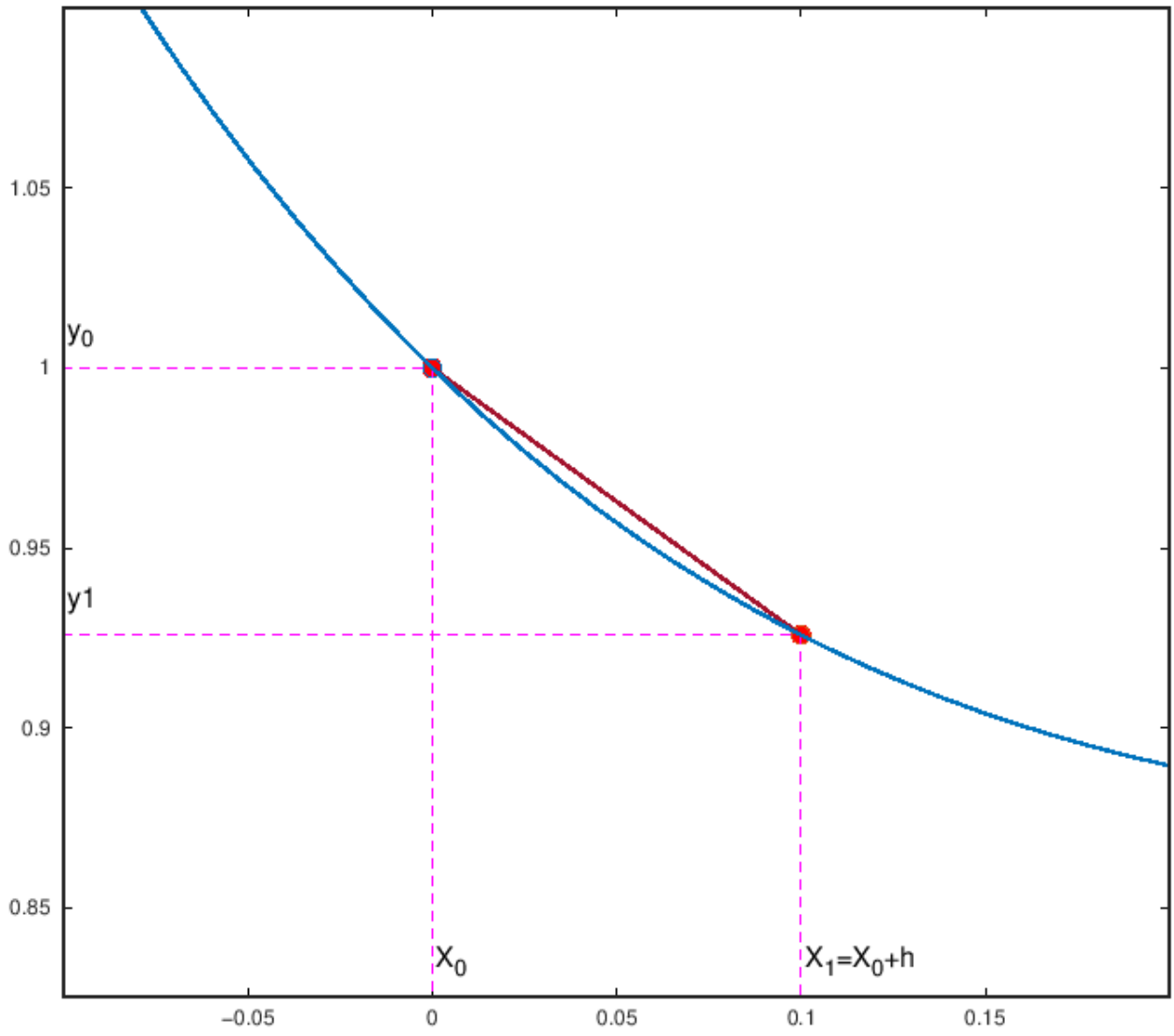
In [64]:
```
plot_rk4_st_5(x0,y0,h,ode)
plot_function([x0-0.1,xn],fun)
```



In [65]:
```
plot_rk4_st_6(x0,y0,h,ode)
plot_function([x0-0.1,xn],fun)
```

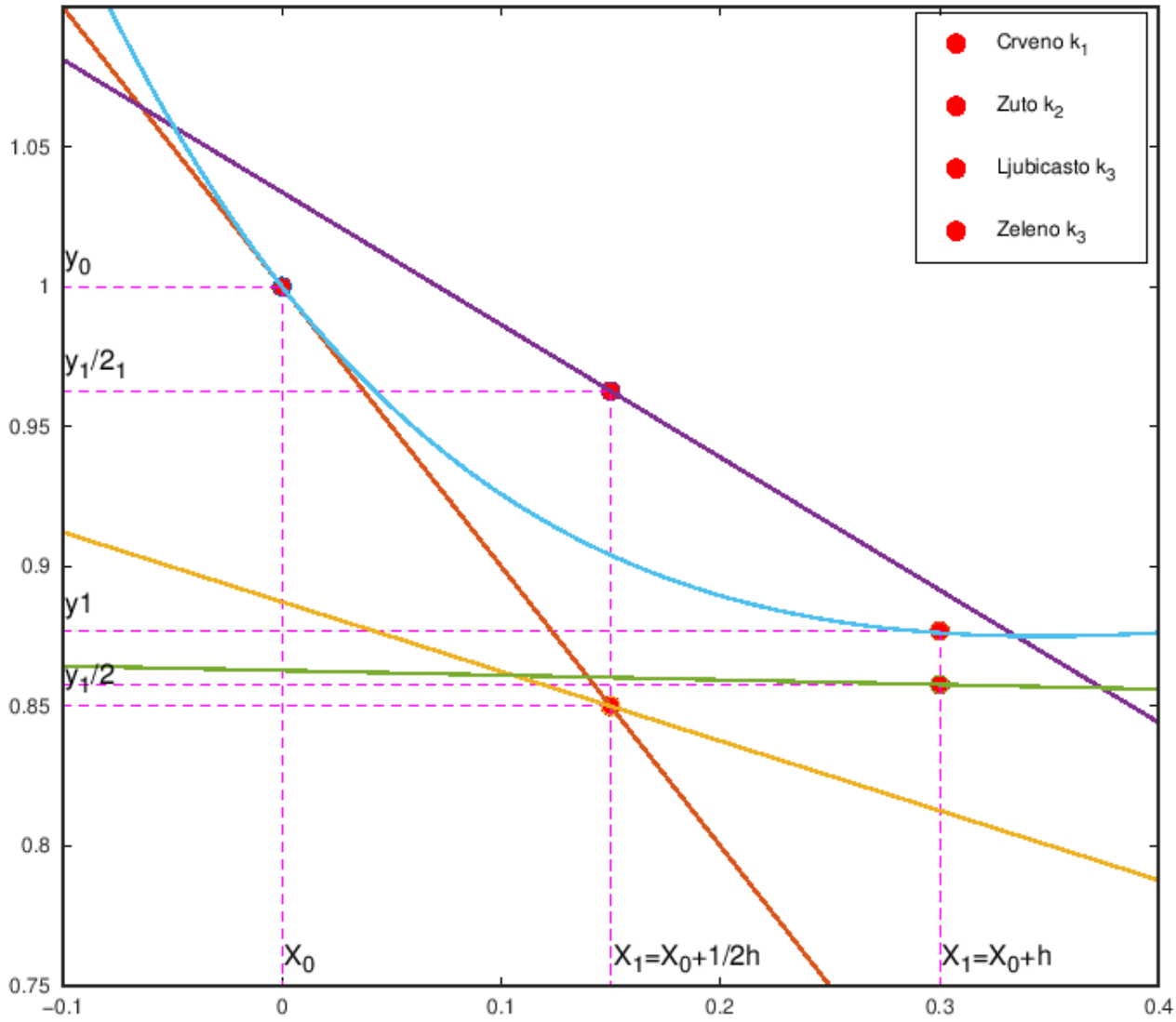


In [66]:
```
plot_rk4_st_7(x0,y0,h,ode)
plot_function([x0-0.1,xn],fun)
```



Prikazaćemo sada sve tangente na jednom grafiku.

In [67]:
```
plot_rk4_st_all(x0,y0,0.3,ode)
plot_function([x0-0.1,xn],fun)
h=legend('Crveno k_1','Zuto k_2', 'Ljubicasto k_3', 'Zeleno k_3',"location", "northeast");
set(h, "fontsize", 12);
```



U nastavku je kod RK4 metoda.

In [68]:
```
function y=RungeKutta4(x0,xn,y0,h,odj)
    x=x0:h:xn;
    n=length(x);
    y=zeros(1,n);
    y(1)=y0;
    for i=2:n
        k1=h*feval(odj,x(i-1),y(i-1));
        k2=h*feval(odj,x(i-1)+1/2*h,y(i-1)+1/2*k1);
        k3=h*feval(odj,x(i-1)+1/2*h,y(i-1)+1/2*k2);
        k4=h*feval(odj,x(i-1)+h,y(i-1)+k3);
        y(i)=y(i-1)+1/6*(k1+2*k2+2*k3+k4);
    end
endfunction
```

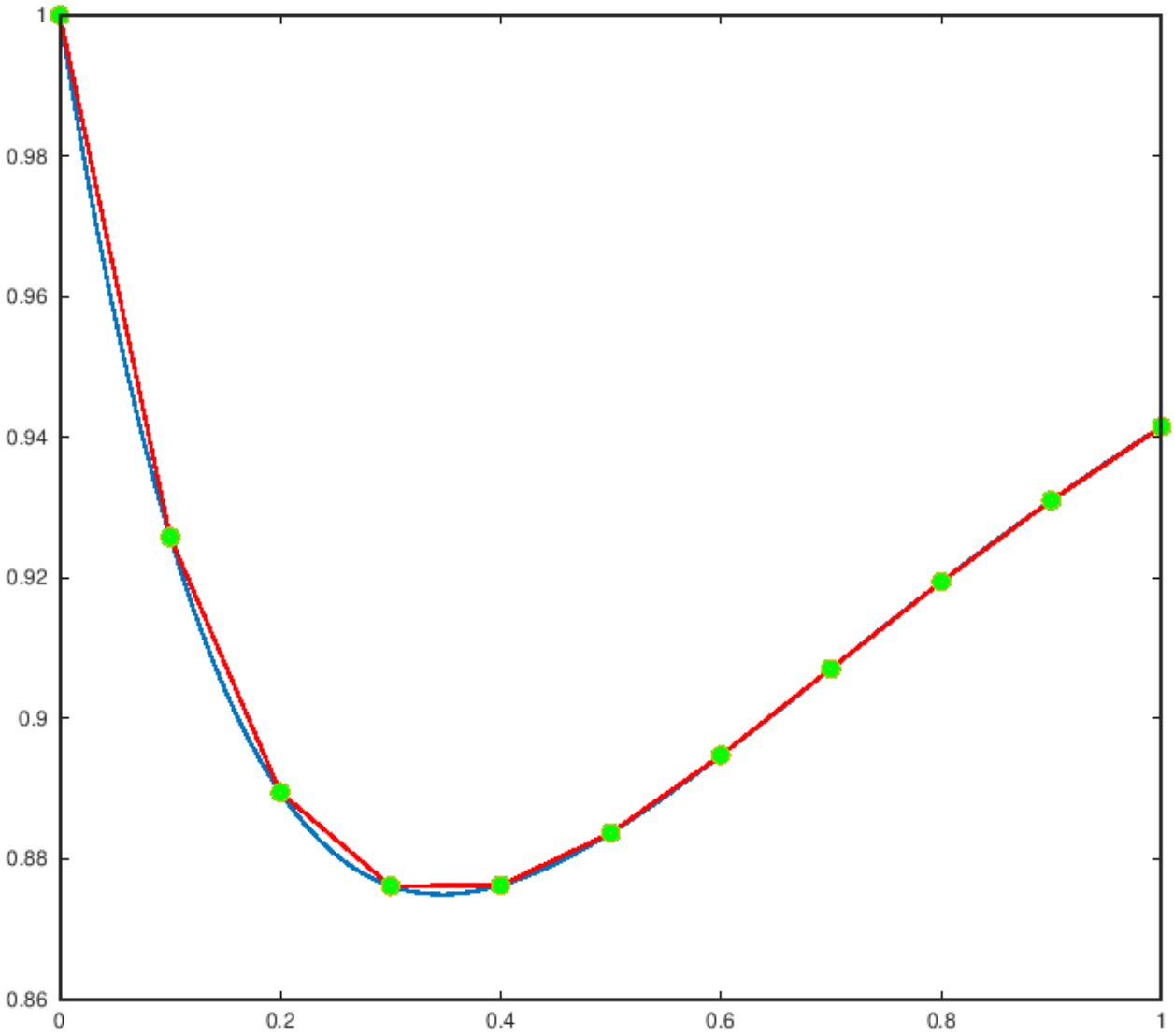
In [69]:

```
ode=@(x,y)2-exp(-4.*x)-2.*y;
x0=0;y0=1;xn=1;h=0.1;
y=RungeKutta4(x0,xn,y0,h,ode)
```

```
y =

 Columns 1 through 8:

   1.00000   0.92580   0.88951   0.87620   0.87629   0.88373   0.89477   0.90711

 Columns 9 through 11:

   0.91944   0.93101   0.94149
```

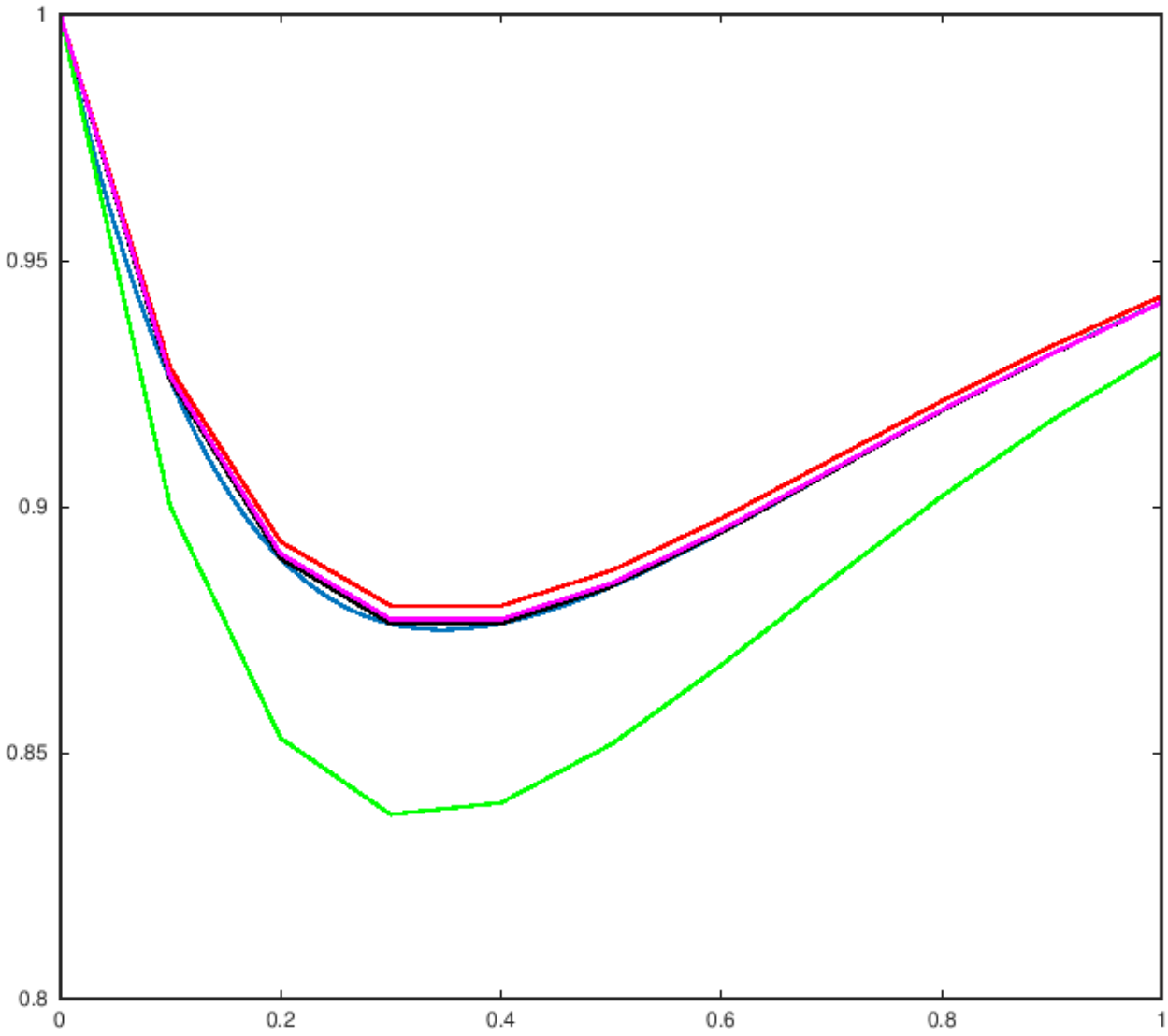
In [70]:
```
p=linterp(x0:h:xn,y)
plot_function([x0,xn],fun);
hold on;
plot(xp,polyval(p,xp),"linewidth", 5,"color", "red");
plot(x0:h:xn,y,'o','markersize', 10,'markerfacecolor','g');
```

```
p =

 Columns 1 through 7:

   0.020845  -0.156902   0.580107  -1.435521   2.706045  -4.099367   4.991713

 Columns 8 through 11:

  -4.665230   2.999751  -0.999951   1.000000
```

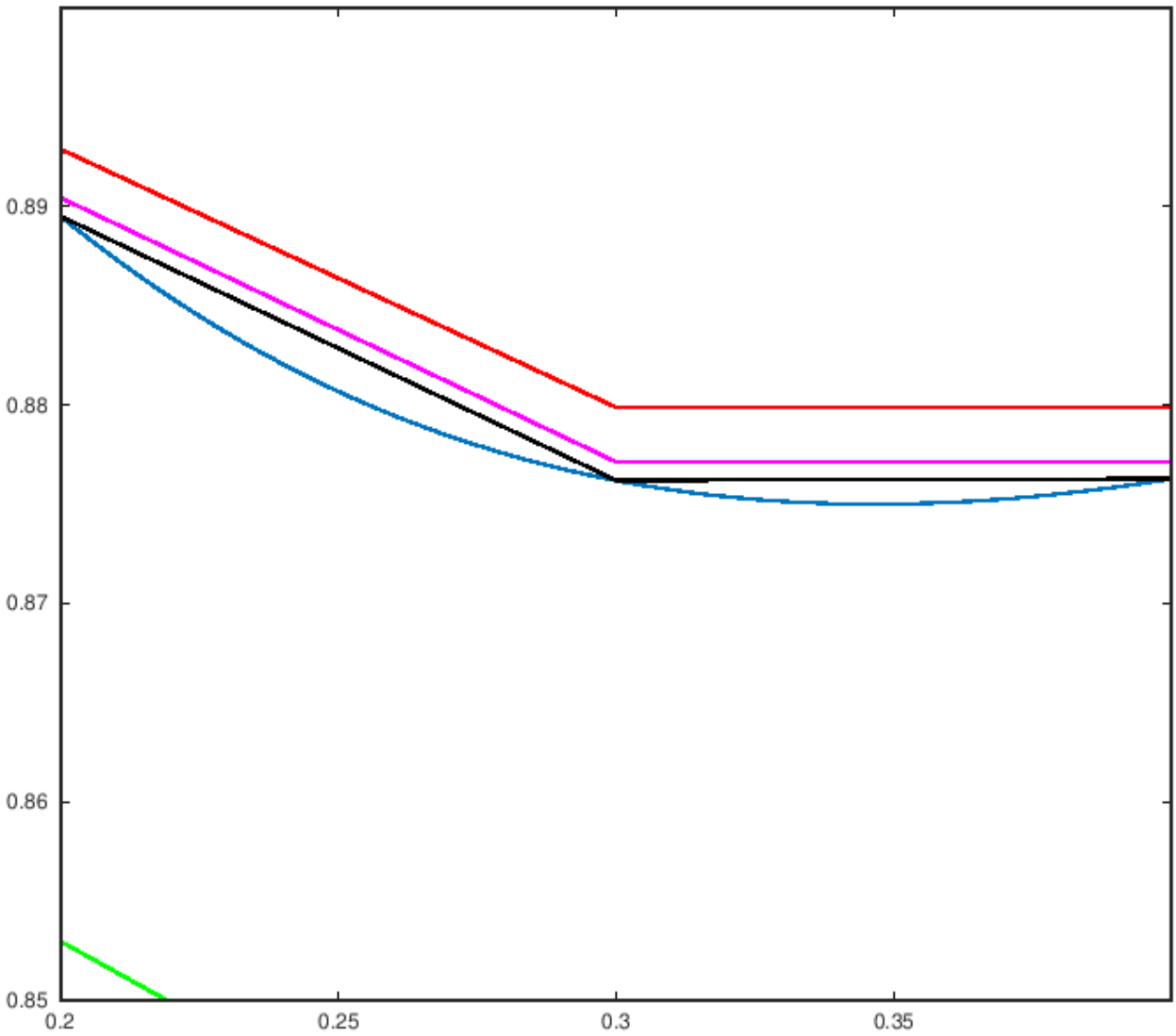


Poredimo Ojlerov metod (zeleno), metod srednje tačke (crveno), Henov metod (magenta) i RK4 metod crno.

In [71]:
```
plot_function([x0,xn],fun);
hold on;
p_ojler=linterp(x0:h:xn,y_ojler);
p_stacka=linterp(x0:h:xn,y_stacka);
p_heun=linterp(x0:h:xn,y_heun);
plot(xp,polyval(p,xp),"linewidth", 5,"color", "black");
plot(xp,polyval(p_ojler,xp),"linewidth", 5,"color", "green");
plot(xp,polyval(p_stacka,xp),"linewidth", 5,"color", "red");
plot(xp,polyval(p_heun,xp),"linewidth", 5,"color", "magenta");
```



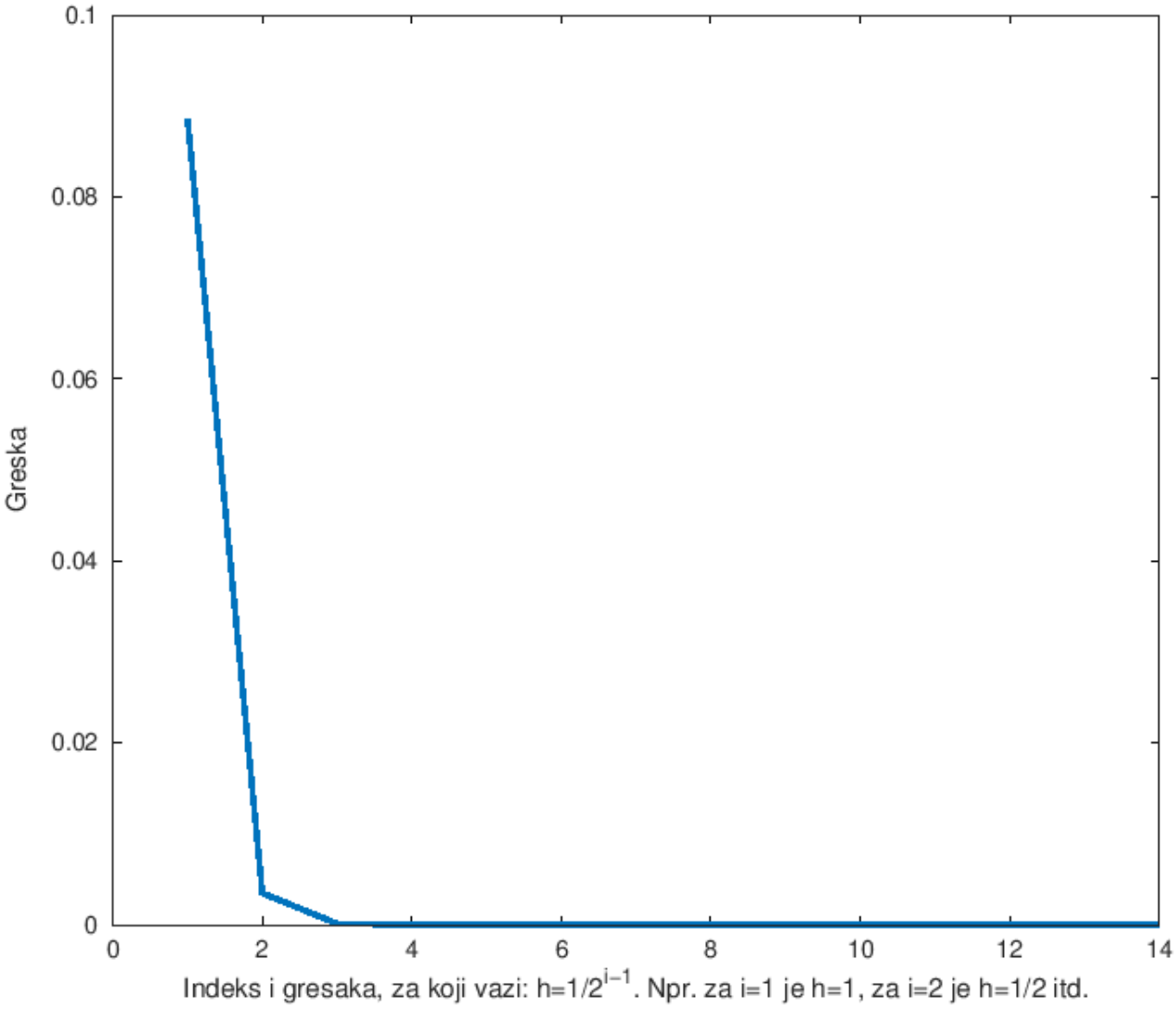
In [72]:
```
plot_function([x0,xn],fun);
hold on;
p_ojler=linterp(x0:h:xn,y_ojler);
p_stacka=linterp(x0:h:xn,y_stacka);
p_heun=linterp(x0:h:xn,y_heun);
plot(xp,polyval(p,xp),"linewidth", 5,"color", "black");
plot(xp,polyval(p_ojler,xp),"linewidth", 5,"color", "green");
plot(xp,polyval(p_stacka,xp),"linewidth", 5,"color", "red");
plot(xp,polyval(p_heun,xp),"linewidth", 5,"color", "magenta");
axis([0.2, 0.4, 0.85,0.9]);
```



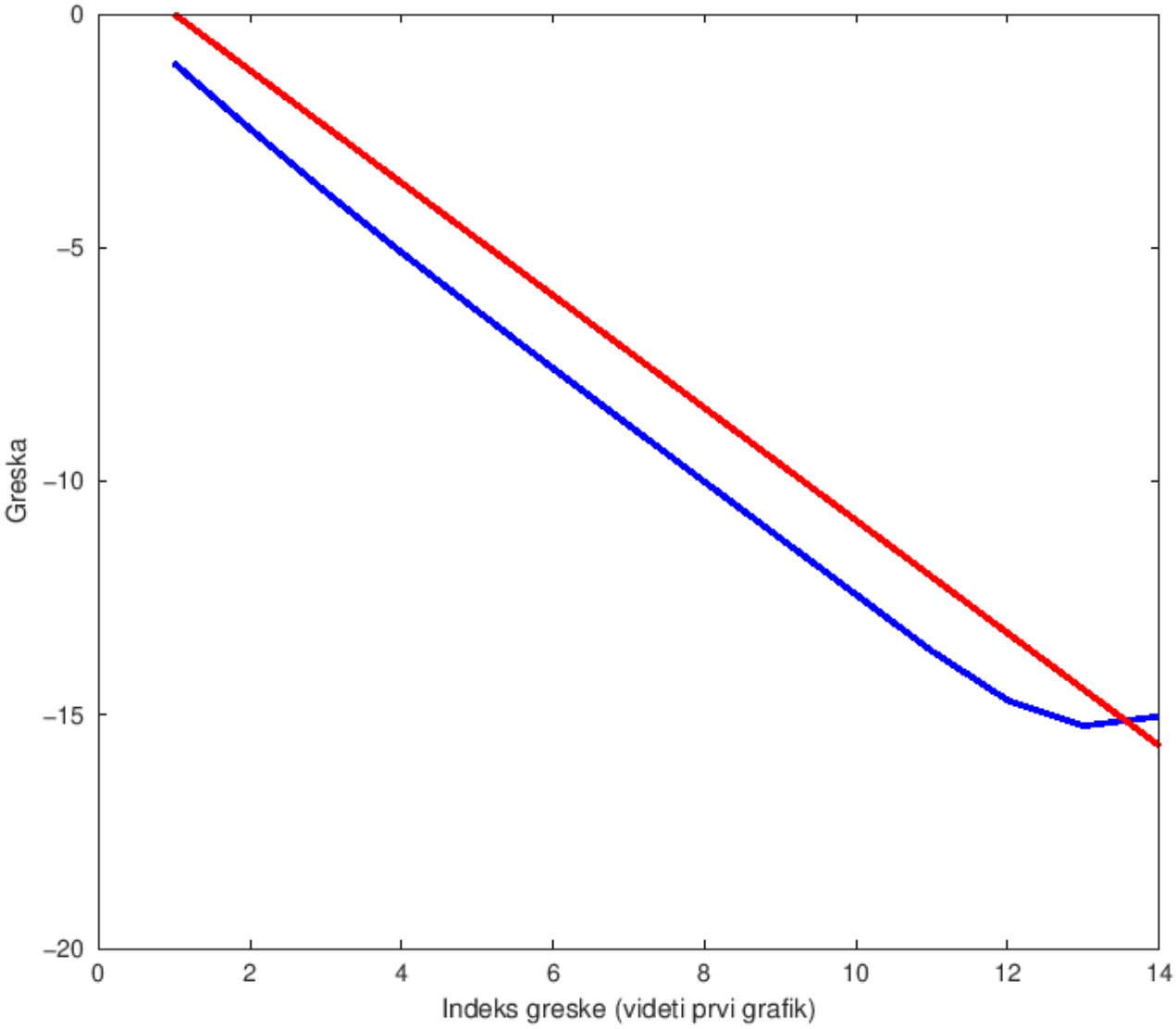
Greška RK4 metoda

Kao što smo već naveli red greške RK4 metoda je $O(h^4)$. U nastavku grafički prikazujemo grešku i poredimo je sa ostalim metodama.

In [73]:
```
[errors,sub_intervals]=calculate_error(x0,xn,y0,ode,fun,0.0001,'RungeKutta4');
```



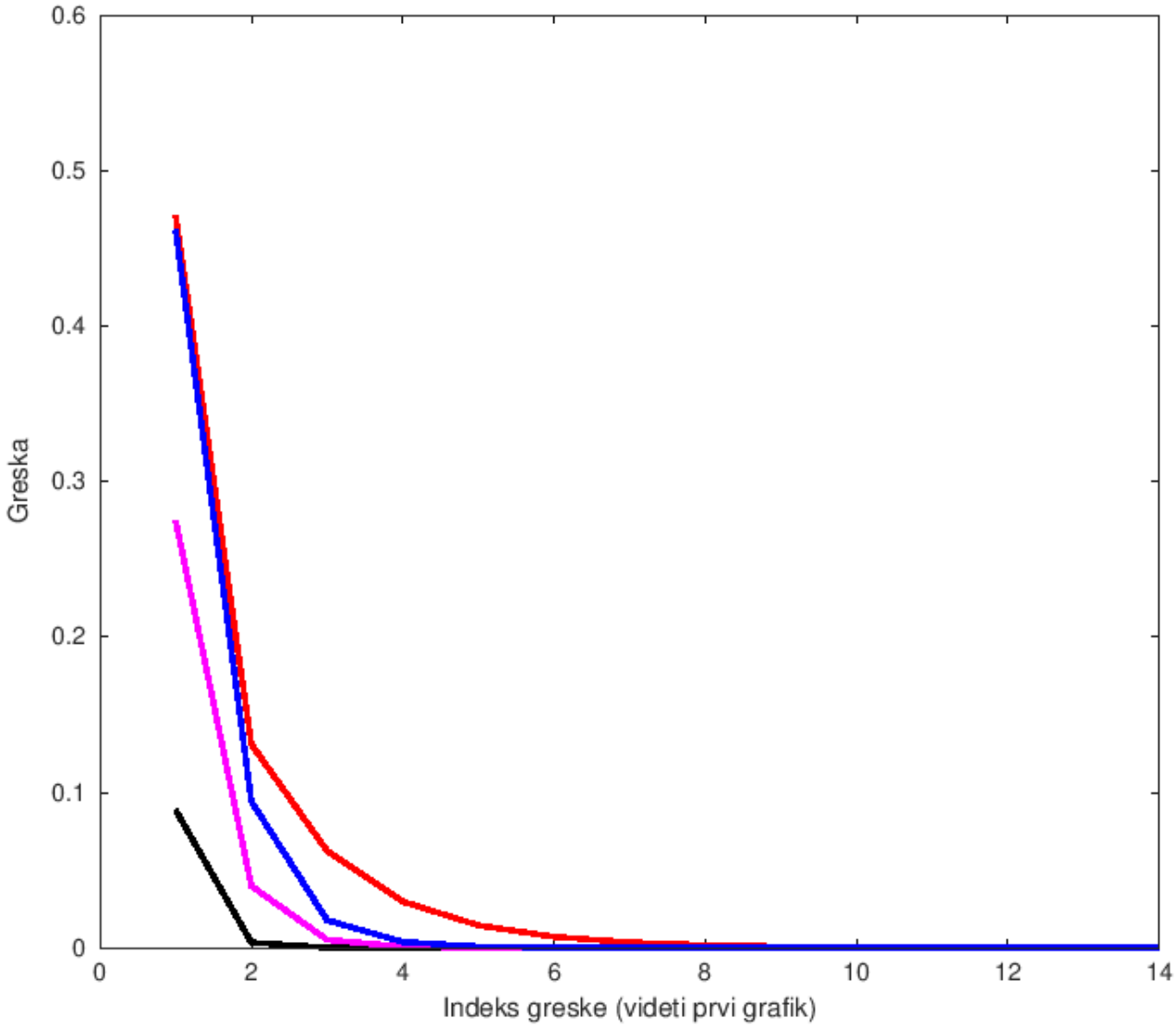
In [74]:
```
plot(1:length(errors),log10(errors),"linewidth",10,"color","blue")
hold on;
plot(1:length(sub_intervals),log10(sub_intervals.^4),"linewidth",10,"color", "red")
xlabel('Indeks greske (videti prvi grafik)');
ylabel('Greska');
set(gca, "fontsize", 14)
```



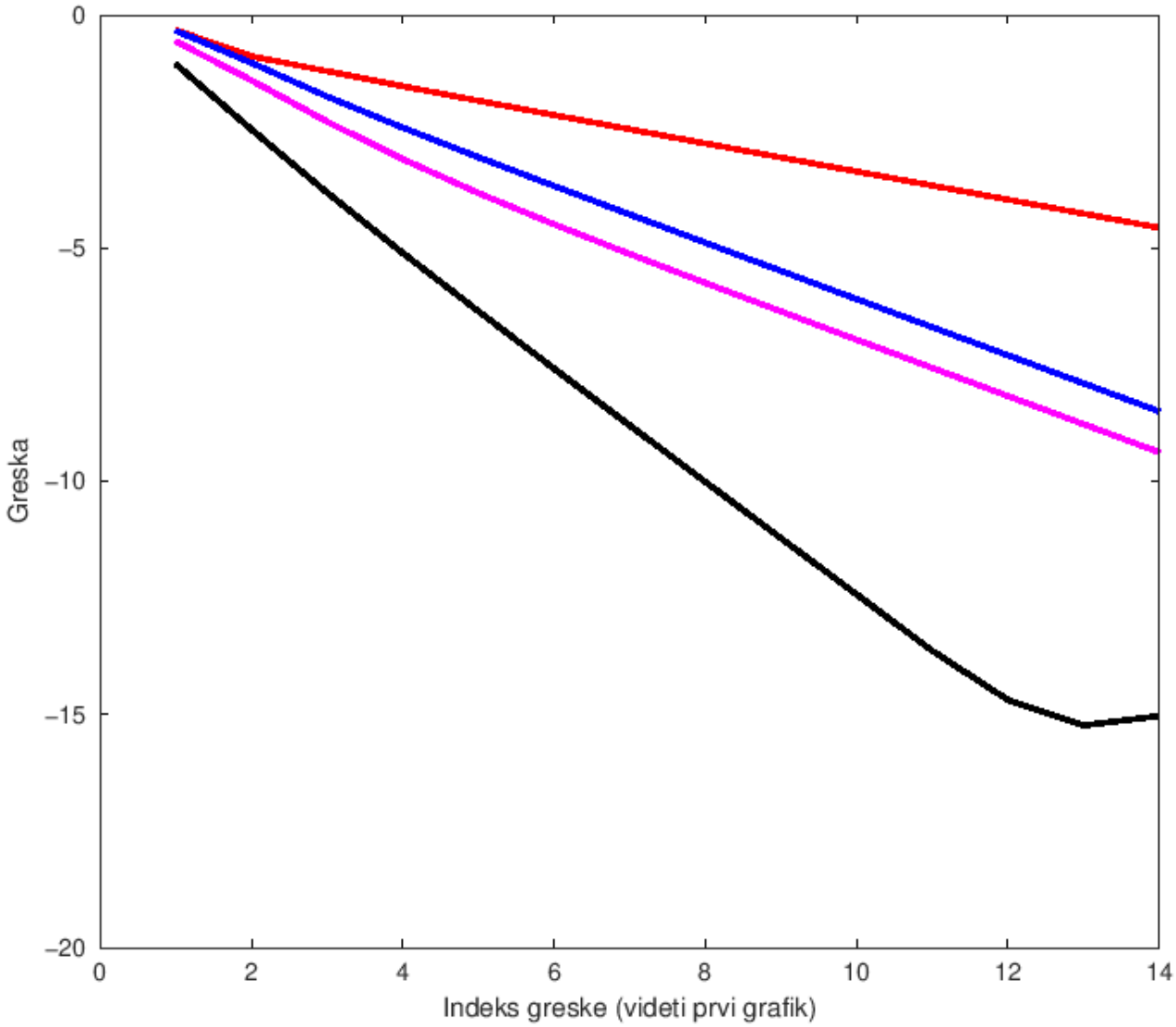
Poredimo sada grešku Ojlerovog metoda (crveno), metoda srednje tačke (plavo), Heunovog metoda (magenta) i RK4 metoda (crno).

In [77]:

```
plot(1:length(errors),errors,"linewidth",10,"color","black")
hold on;
plot(1:length(errors),errors_heun,"linewidth",10,"color","magenta")
plot(1:length(errors),errors_ojler,"linewidth",10,"color", "red")
plot(1:length(errors),errors_stacka,"linewidth",10,"color","blue")
xlabel('Indeks greske (videti prvi grafik)');
ylabel('Greska');
set(gca, "fontsize", 14)
```



In [78]:
```
plot(1:length(errors),log10(errors),"linewidth",10,"color","black")
hold on;
plot(1:length(errors),log10(errors_heun),"linewidth",10,"color","magenta")
plot(1:length(errors),log10(errors_ojler),"linewidth",10,"color", "red")
plot(1:length(errors),log10(errors_stacka),"linewidth",10,"color","blue")
xlabel('Indeks greske (videti prvi grafik)');
ylabel('Greska');
set(gca, "fontsize", 14)
```



In [ ]: