### **Example 3.2.4 Nonlinear Heat Transfer**

> restart:

> with(plots):

Enter the governing equation:

> eq:=diff(y(x),x$2)-(1+0.1\*y(x))\*y(x);



The sensitivity equation is developed by treating y as a function of x and α:

> eqalpha:=subs(y(x)=Y(x,alpha),eq);



The governing equation for the Jacobian is obtained by differentiating the governing equation with respect to α:



> eqalpha:=diff(eqalpha,alpha);



A new variable, y2(x) is used to present the Jacobian



> eqalpha:=subs(diff(Y(x,alpha),alpha)=y2(x),eqalpha);



> eqalpha:=subs(Y(x,alpha)=y(x),eqalpha);



The variables are stored in vars:

> vars:=(y(x),y2(x));



The original governing equation and the sensitivity equation are stored in eqs:

> eqs:=(eq,eqalpha);



The initial value for α is given here:

> alpha0:=0.5;



The initial conditions are stored in ICs:

> ICs:=(y(0)=alpha0,D(y)(0)=0,y2(0)=1,D(y2)(0)=0);



Next the numerical solution is obtained and stored in sol:

> sol:=dsolve({eqs,ICs},{vars},type=numeric);



The solution is evaluated at x = 1:

> sol(1);



The predicted value of y is stored in ypred:

> ypred:=rhs(sol(1)[2]);



The predicted value for the Jacobian is stored in y2pred:

> y2pred:=rhs(sol(1)[4]);



The new value for α is obtained as:

> alpha1:=alpha0+(1-ypred)/y2pred;



The error is calculated based on the value of α:

> err:=alpha1-alpha0;



The new value of α is then assigned to α0 for the next iteration.

> alpha0:=alpha1;



A program is written to update the values of α until the error is > 1e - 6. One can set stricter tolerance limits for higher accuracy.

> k:=1;



> while err>1e-6 do

> ICs:=(y(0)=alpha0,D(y)(0)=0,y2(0)=1,D(y2)(0)=0);

> sol:=dsolve({eqs,ICs},{vars},type=numeric);

> ypred:=rhs(sol(1)[2]);

> y2pred:=rhs(sol(1)[4]);

> alpha1:=alpha0+(1-ypred)/y2pred;

> err:=abs(alpha1-alpha0);

> alpha0:=alpha1;k:=k+1;

> end;



> k;



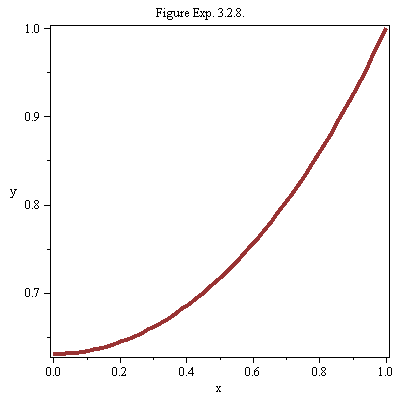
Three iterations were required for this problem. The error in α is found to be:

> err;



The solution obtained is then plotted:

> odeplot(sol,[x,y(x)],0..1,axes=boxed,color=brown,title="Figure Exp. 3.2.8.",thickness=4);



For this boundary value problem, it takes only three iterations for the solution to converge. Depending on the problem and the initial guess provided, the program might take any number of iterations to converge. In addition, for updating α (equation 3.2.11) the Jacobian in the denominator might approach zero for certain problems. Sometimes it is necessary to scale the update of α using the following relations:

αnew= αold (3.2.16)



where ρ is the number between 0 and 1. The lower the value of ρ the higher is the number of iterations required for convergence. The value of ρ depends on the problem. This is illustrated in the next example.

>