Lezione 10 (parte prima)

Enrico Bertolazzi

```
> with(plots) :
Warning, the name changecoords has been redefined
```

```
> RK4 := proc (t0, # tempo iniziale
               tf, # tempo finale
               x0, # dato iniziale
               y0, # dato iniziale
               h,  # passo di avanzamento
               f,  #
               g)  #
    local i, L, ti, xi, yi, ti1, xi1, yi1,
          KX1, KX2, KX3, KX4,
          KY1, KY2, KY3, KY4 ;
    # inizializzazione
    i  := 0 ;
    L  := [[x0,y0]] ;
    ti := t0 ;
    xi := x0 ;
    yi := y0 ;
    # ciclo
    while ti < tf do
      # calcolo K1
      KX1 := evalf(h*f(ti,xi,yi)) ;
      KY1 := evalf(h*g(ti,xi,yi)) ;
      # calcolo K2
      KX2 := evalf(h*f(ti+h/2,xi+KX1/2,yi+KY1/2)) ;
      KY2 := evalf(h*g(ti+h/2,xi+KX1/2,yi+KY1/2)) ;
      # calcolo K3
      KX3 := evalf(h*f(ti+h/2,xi+KX2/2,yi+KY2/2)) ;
      KY3 := evalf(h*g(ti+h/2,xi+KX2/2,yi+KY2/2)) ;
      # calcolo K4
      KX4 := evalf(h*f(ti+h,xi+KX3,yi+KY3)) ;
      KY4 := evalf(h*g(ti+h,xi+KX3,yi+KY3)) ;
      # calcolo prossimo punto
      ti1 := evalf(ti+h) ;
      xi1 := evalf(xi + (KX1+2*KX2+2*KX3+KX4)/6 ) ;
      yi1 := evalf(yi + (KY1+2*KY2+2*KY3+KY4)/6 ) ;
      # aggiorno lista con soluzioni
      L   := [op(L),[xi1,yi1]] ;
      # copia i valori calcolati per il prossimo passo
      ti := ti1 ;
      xi := xi1 ;
      yi := yi1 ;
```

```
    end ;
    return L ;
  end ;
```

$$RK4 := \mathbf{proc}(t0, tf, x0, y0, h, f, g) \tag{1}$$

```
    local i, L, ti, xi, yi, ti1, xi1, yi1, KX1, KX2, KX3, KX4, KY1, KY2, KY3, KY4;
    i := 0;
    L := [[x0, y0]];
    ti := t0;
    xi := x0;
    yi := y0;
    while ti < tf do
        KX1 := evalf(h*f(ti, xi, yi));
        KY1 := evalf(h*g(ti, xi, yi));
        KX2 := evalf(h*f(ti + 1/2*h, xi + 1/2*KX1, yi + 1/2*KY1));
        KY2 := evalf(h*g(ti + 1/2*h, xi + 1/2*KX1, yi + 1/2*KY1));
        KX3 := evalf(h*f(ti + 1/2*h, xi + 1/2*KX2, yi + 1/2*KY2));
        KY3 := evalf(h*g(ti + 1/2*h, xi + 1/2*KX2, yi + 1/2*KY2));
        KX4 := evalf(h*f(ti + h, xi + KX3, yi + KY3));
        KY4 := evalf(h*g(ti + h, xi + KX3, yi + KY3));
        ti1 := evalf(ti + h);
        xi1 := evalf(xi + 1/6*KX1 + 1/3*KX2 + 1/3*KX3 + 1/6*KX4);
        yi1 := evalf(yi + 1/6*KY1 + 1/3*KY2 + 1/3*KY3 + 1/6*KY4);
        L := [op(L), [xi1, yi1]];
        ti := ti1;
        xi := xi1;
        yi := yi1
    end do;
    return L
end proc
```

```
> #
  # definizione del problema
  # con soluzione esatta sul cerchio
  #
  fx := (t,x,y) -> y ;
  fy := (t,x,y) -> -x ;
```

```
t0 := 0 ;
tf := 5 ;
x0 := 0 ;
y0 := 1 ;
```

$$fx := (t, x, y) \rightarrow y \qquad (2)$$

$$fy := (t, x, y) \rightarrow -x$$

$$t0 := 0$$

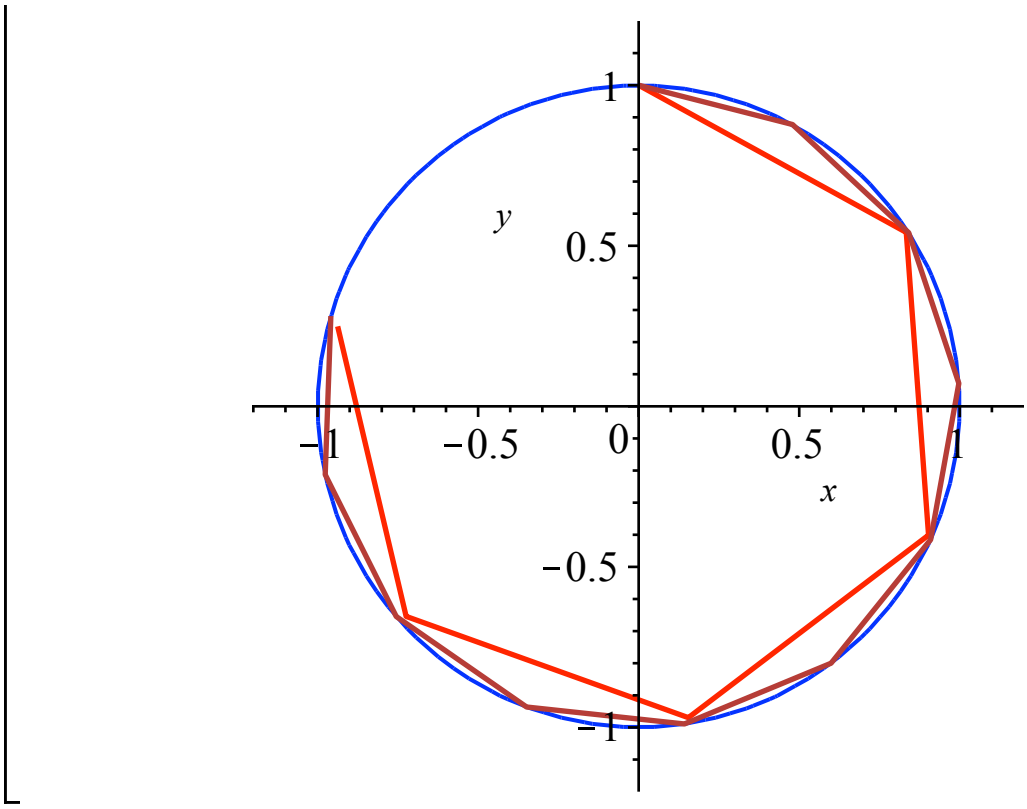
$$tf := 5$$

$$x0 := 0$$

$$y0 := 1$$

```
> # salvo in C il "plot" del cerchio
  C := implicitplot(x^2+y^2-1,
                    x=-1.2..1.2,y=-1.2..1.2,
                    color=blue,thickness=1) :

> # prima soluzione con h=1 (red)
  h := 1 :
  L1 := RK4(t0,tf,x0,y0,h,fx,fy) :
  B1 := plot(L1,-1.2..1.2,-1.2..1.2,
             color=red,thickness=2) :
> # seconda soluzione con h=0.5 (brown)
  h := 0.5 :
  L2 := RK4(t0,tf,x0,y0,h,fx,fy) :
  B2 := plot(L2,-1.2..1.2,-1.2..1.2,
             color=brown,thickness=2) :
> # combino le soluzioni in un unico grafico
  display({C,B1,B2},scaling=CONSTRAINED) ;
```


1
y
0.5
−1
−0.5
0
0.5
1
x
−0.5
−1