Runge Kutta di Dormand Prince

Ho realizzato anche la versione di questo algoritmo in Javascript e nel linguaggio di Maxima (maximese) usando come interfaccia grafica wxMaxima.
Ora faccio la versione per *Mathematica* ...

In[1]:=
```
DateString[]
```

Out[1]=
```
Fri 26 Jul 2013 11:03:46
```

http://en.wikipedia.org/wiki/Dormand% E2 %80 %93 Prince_method

http://en.wikipedia.org/wiki/Butcher_tableau# Explicit_Runge.E2.80.93Kutta_methods

---

La mia variante a coefficienti interi

Il nucleo dell'algoritmo di Dormand Prince scritto in mathematichese è questo:

In[2]:=
```
nucleode[] := Block[{},
 ha21 = 34 272 564 480 * h;
 k2 = fder[x + 34 272 564 480 * h, y + ha21 * k1, costa];
 ha31 = 12 852 211 680 * h;
 ha32 = 38 556 635 040 * h;
 k3 = fder[x + 51 408 846 720 * h, y + ha31 * k1 + ha32 * k2, costa];
 ha41 = 167 554 759 680 * h;
 ha42 = -639 754 536 960 * h;
 ha43 = 609 290 035 200 * h;
 k4 = fder[x + 137 090 257 920 * h, y + ha41 * k1 + ha42 * k2 + ha43 * k3, costa];
 ha51 = 505 965 644 800 * h;
 ha52 = -1 987 087 872 000 * h;
 ha53 = 1 683 278 643 200 * h;
 ha54 = -49 833 907 200 * h;
 k5 = fder[x + 152 322 508 800 * h, y + ha51 * k1 + ha52 * k2 +
    ha53 * k3 + ha54 * k4, costa];
 ha61 = 487 745 760 600 * h;
 ha62 = -1 843 448 544 000 * h;
 ha63 = 1 526 229 734 400 * h;
 ha64 = 47 708 967 600 * h;
 ha65 = -46 873 096 200 * h;
 hc6 = 171 362 822 400 * h;
 k6 = fder[x + hc6, y + ha61 * k1 + ha62 * k2 + ha63 * k3 +
   ha64 * k4 + ha65 * k5, costa];
 ha71 = 15 619 007 250 * h;
 ha73 = 76 982 400 000 * h;
 ha74 = 111 564 337 500 * h;
 ha75 = -55 243 291 950 * h;
 ha76 = 22 440 369 600 * h;
 k7 = fder[x + 171 362 822 400 * h,
  y + ha71 * k1 + ha73 * k3 + ha74 * k4 + ha75 * k5 + ha76 * k6, costa];
 ];
```

Ovviamente il presupposto è che la funzione nucleode[ ] venga usata all'interno di una opportuna funzione in modo che trovi definito il valore di x, di y e di h etc...

La funzione base da usare è questa:

In[3]:=
```
odedopri[x0_, y0_, x1_, costa_, peso_, tol_, maxiter_] :=
  Block[{ha21, ha31, ha32, ha41, ha42, ha43, ha51, ha52, ha53, ha54,
    ha61, ha62, ha63, ha64, ha65, ha71, ha73, ha74, ha75, ha76, hc6,
    e1, e3, e4, e5, e6, e7, i, j, ifa, x, y, ny, h, hmax, hmin,
    k1, k2, k3, k4, k5, k6, k7, verrore, erromax, avanti, amen},
   x = x0; y = y0; ny = Length[y];
   h = 10 * (x1 - x0) / (maxiter * 171 362 822 400);
   hmax = h;  hmin = h / 100;
   erromax = 0;
   amen = False;
   e1 = 211 228 479; e3 = -728 766 720;
   e4 = 6 336 854 370; e5 = -8 716 163 841;
   e6 = 7 180 918 272; e7 = -4 284 070 560;
   (* *)
   k1 = fder[x, y, costa];
   For[i = 1, maxiter ≥ i, i++, ifa = i;
    nucleode[];
    verrore = peso * Abs[e1 * k1 + e3 * k3 + e4 * k4 + e5 * k5 + e6 * k6 + e7 * k7];
    erromax = 0;
    For[j = 1, ny ≥ j, j++, If[verrore[[j]] > erromax, erromax = verrore[[j]] ;];
    ];
    If[tol > erromax / 171 362 822 400 , avanti = True,
       If [ hc6 > hmin, h = h / 2; avanti = False, avanti = True; amen = False];
    ];
    If[ avanti, x = x + hc6;
     y = y + ha71 * k1 + ha73 * k3 + ha74 * k4 + ha75 * k5 + ha76 * k6;
     k1 = k7;
     If[amen , Break[]];
     If[hmax > h, h = 105 * h / 100];];
    If[x + h * 171 362 822 400 > x1, h = (x1 - x) / 171 362 822 400; amen = True];
   ];
   {amen, x, y, ifa}];
```

Qui definisco la fder[x,y,c] varie volte ...
In *Mathematica* è ammesso fare diverse definizioni della stessa funzione per cui il nome fder viene qui usato come nome convenzionale della funzione usata dalla funzione odedopri[...].

In[4]:=
```
(* Qui la soluzione è {x,Exp[x]} *)
fder[x_, y_, c_] := Block[{rr}, rr = 0.5 * Exp[x]; {1, rr + y[[2]] / 2}];
```

In[5]:=
```
ini$fder[] := Block[{}, x0 = 0.0; x1 = 1.0; y0 = {0.0, 1.0}; costa = {1.0};
   peso = Table[1.0, {i, Length[y0]}];
   tollero = 1.0 × 10^-13;
   maxiter = 2000;
   Print["Inizializzazione n. 1 eseguita !"];];
ini$fder[];
```

```
Inizializzazione n. 1 eseguita !
```

Calcola la soluzione....

In[7]:=
```
ris = odedopri[x0, y0, x1, costa, peso, tollero, maxiter];
Print["Primo caso test ", ris];
```

```
Primo caso test {True, 1., {1., 2.71828}, 312}
```

Ora cambio la definizione della fder per verificare l'esistenza di questa facoltà e poi risolvo il sistema ottenendo, ovviamente risultati diversi...

In[9]:=
```
(* Qui la soluzione è {x,Exp[2*x]} *)
fder[x_, y_, c_] := Block[{rr}, rr = 0.5 * Exp[2 * x]; {1, rr + 1.5 * y[[2]]}];
```

In[10]:=
```
ris = odedopri[x0, y0, x1, costa, peso, tollero, maxiter];
Print["Variante del primo caso test ", ris];
```

```
Variante del primo caso test {True, 1., {1., 7.38906}, 1133}
```

Un problemino che richiede funzioni trigonometriche...

In[12]:=
```
(* Secondo esempio usabile dalla funzione fder.
   Qui la soluzione è y=Sin[x],z=2*Cos[x] *)
fder[x_, v_, c_] := Block[{y, z}, y = v[[1]]; z = v[[2]]; {z * 0.5, -2 * y}];
ini$fder[] := Block[{}, x0 = 0; x1 = Pi * 1.5; y0 = {0, 2.0}; costa = {1.0};
   peso = Table[1.0, {i, Length[y0]}]; tollero = 1 × 10^-13;
   maxiter = 2500;
   Print["Inizializzazione n. 2 eseguita !"];];
(* Dovendo arrivare a 3 Pi/2 la soluzione è {-1,0} *)
ini$fder[];
```

```
Inizializzazione n. 2 eseguita !
```

In[15]:=
```
ris = odedopri[x0, y0, x1, costa, peso, tollero, maxiter];
Print["Secondo caso test ", ris];
```

```
Secondo caso test {True, 4.71239, {-1., -1.62039 × 10^-14}, 2274}
```

Qui calcolo il moto di un pianeta attorno al Sole. La lista c contiene le costanti usate nel calcolo.

In[17]:=
```
fder[t_, v_, c_] := Block[{x, y, vx, vy, mu,
     rq, rr, den, xd, yd, vxd, vyd},
   x = v[[1]]; y = v[[2]]; vx = v[[3]]; vy = v[[4]];
   mu = c[[1]]; xd = vx; yd = vy; rq = x * x + y * y;
   rr = Sqrt[rq]; den = 1 / (rr * rq);
   vxd = -mu * x * den; vyd = -mu * y * den;
   {xd, yd, vxd, vyd}];
ini$fder[] := Block[{vm, vp, va, ec},
   costa = {1.327581 × 10^20, 149.61 × 10^9, 30500.0};
   vm = Sqrt[costa[[1]] / costa[[2]]]; If[ costa[[3]] > vm ,
    vp = costa[[3]]; va = vm * vm / vp, va = costa[[3]]; vp = vm * vm / va];
   ec = (vp - va) / (vp + va); x0 = 0; x1 = 2 * Pi * costa[[2]] / vm;
   y0 = {costa[[2]] * (1 - ec), 0, 0, vp};
   peso = {1.0 × 10^-9, 1.0 × 10^-9, 1.0 × 10^-4, 1.0 × 10^-4};
   tollero = 1 × 10^-15;
   maxiter = 1500;
   Print["Inizializzazione n. 3 eseguita !"];];
ini$fder[];
```

```
Inizializzazione n. 3 eseguita !
```

Dopo un intero pseudo_anno dovrebbe ritornare alla posizione di partenza ossia y0 :

In[20]:=
```
Print["Posizione di partenza della pseudo Terra ", y0];
```

```
Posizione di partenza della pseudo Terra {1.4608 × 10^11, 0, 0, 30500.}
```

In[21]:=
```
ris = odedopri[x0, y0, x1, costa, peso, tollero, maxiter];
Print["Terzo caso test ", ris];
```

```
Terzo caso test
 {True, 3.15566 × 10^7, {1.4608 × 10^11, -0.488263, 9.98766 × 10^-8, 30500.}, 641}
```

Un sistema a tre corpi del tipo Sole-Terra-Luna trattato però in soltanto due dimensioni, per non complicare troppo l'esempio.
Il questo caso, nel moto piano, le variabili diventano otto e non esiste più una soluzione analitica.
Il valore ts rappresenta il rapporto tra la massa del Sole e quello della Terra mentre il valore ls rappresenta il rapporto tra la massa della Luna e quella del Sole.

Come istante di inizio scelgo quello in cui la Terra è al perielio e dunque l'ascissa è la distanza Terra-Sole mentre l'ordinata è nulla.
La Luna ha la stessa ascissa della Terra ma ha l'ordinata decrementata della distanza media tra Terra e Luna ossia 360e6 metri mentre la velocità è in allontanamento dal Sole è di circa 1100 m/s mentre la velocità tangente all'orbita è la stessa della Terra ossia circa 30500 m/s.

In[23]:=
```
fder[t_, v_, c_] := Block[{mu, ts, xt, yt, vxt, vyt, rqt, rrt, dent,
     xtd, ytd, xld, yld, vxtd, vytd, vxld, vyld, dentl, denlt,
     ls, xl, yl, vxl, vyl, rql, rqtl, rrl, rrtl, denl},
    xt = v[[1]]; yt = v[[2]]; xl = v[[3]]; yl = v[[4]];
    vxt = v[[5]]; vyt = v[[6]]; vxl = v[[7]]; vyl = v[[8]];
    mu = c[[1]]; ts = c[[2]]; ls = c[[3]];
    xtd = vxt;
    ytd = vyt;
    xld = vxl;
    yld = vyl;
    rqt = xt * xt + yt * yt; rrt = Sqrt[rqt];
    dent = 1 / (rrt * rqt);
    rql = xl * xl + yl * yl;  rrl = Sqrt[rql];
    denl = 1 / (rrl * rql);
    rqtl = (xt - xl) * (xt - xl) + (yt - yl) * (yt - yl); rrtl = Sqrt[rqtl];
    dentl = ls / (rqtl * rrtl); denlt = ts / (rqtl * rrtl);
    vxtd = -mu * (xt * dent + (xt - xl) * dentl);
    vytd = -mu * (yt * dent + (yt - yl) * dentl);
    vxld = -mu * (xl * denl + (xl - xt) * denlt);
    vyld = -mu * (yl * denl + (yl - yt) * denlt);
    {xtd, ytd, xld, yld, vxtd, vytd, vxld, vyld}];
ini$fder[] := Block[{vm, vp, va, ec},
    costa = {1.327581 × 10^20, 1 / 333 000.1, 1 / (333 000.1 * 80),
      149.61 × 10^9, 30 500.0}; vm = Sqrt[costa[[1]] / costa[[4]]];
    If[ costa[[5]] > vm ,  vp = costa[[5]]; va = vm * vm / vp ,
     va = costa[[5]]; vp = vm * vm / va];
    ec = (vp - va) / (vp + va);
    Print["Eccentricità ", ec, "\nVelocità media ", vm];
    x0 = 0;
    x1 = 2 * Pi * costa[[4]] / vm;
    y0 = {costa[[4]] * (1 - ec), 0, costa[[4]] * (1 - ec),
      -360 * 10^6, 0, vp, 1100.0, vp};
    peso = {1. × 10^-9, 1. × 10^-9, 1. × 10^-9, 1. × 10^-9,
      1. × 10^-4, 1. × 10^-4, 1. × 10^-4, 1. × 10^-4};
    tollero = 1. × 10^-15;
    maxiter = 2000;
    {x0, y0, x1}];
ini$fder[];
```

```
Eccentricità 0.0235963
Velocità media 29 788.6
```

In[26]:=
```
ris1 = odedopri[x0, y0, x1, costa, peso, tollero * 100, maxiter];
Print["Quarto caso test precisione bassa ", ris1];
```

```
Quarto caso test precisione bassa
 {True, 3.15566 × 10^7, {1.46085 × 10^11, -3.67465 × 10^6, 1.45667 × 10^11,
   -8.02539 × 10^7, 10.5156, 30 511.2, 261.616, 29 601.4}, 1094}
```

In[28]:=
```
ris2 = odedopri[x0, y0, x1, costa, peso, tollero * 10, maxiter];
Print["Quarto caso test precisione media ", ris2];
```

```
Quarto caso test precisione media
 {True, 3.15566 × 10^7, {1.46085 × 10^11, -3.67461 × 10^6, 1.45667 × 10^11,
   -8.02567 × 10^7, 10.5155, 30 511.2, 261.622, 29 601.4}, 1952}
```

In[30]:=
```
ris3 = odedopri[x0, y0, x1, costa, peso, tollero, maxiter * 3];
Print["Quarto caso test precisione alta ", ris3];
```

```
Quarto caso test precisione alta
 {True, 3.15566 × 10^7, {1.46085 × 10^11, -3.67461 × 10^6, 1.45667 × 10^11,
   -8.02569 × 10^7, 10.5155, 30511.2, 261.623, 29601.4}, 3459}
```

Il problema dei tre corpi non ammette una soluzione analitica ma queste tre prove a diverso livello di precisione mi sembra che dimostrino che la soluzione trovata è ... ragionevole.

Conclusione

Credo che funzioni... **Ora va APPLICATO A CASI REALMENTE INTERESSANTI...**

Presto lo userò come caso di confronto tra come l'algoritmo va programmato per wxMaxima e per *Mathematica.* Lo metterò qui:

http://www.elegio.it/max/

Dandogli un certo risalto. Ovviamente la soluzione delle equazioni differenziali alle derivate ordinarie è un "pezzo forte" di qualsiasi programma di calcolo matematico, numerico o simbolico che sia...

A questo argomento ho dedicato molto impegno (logicamente). Chi ha il gratuito software per leggere il "Computable Document Format" può guardare questa mia pagina :

http://www.elegio.it/cdf/kepleriano-preciso-2.cdf.html

E queste vecchie pagine:

http://www.elegio.it/mc2/rk/

Anche se non è più considerato IL MIGLIORE suggerisco di dare una occhiata a questo documento che ho salvato sul mio sito:

http://www.elegio.it/mc2/rk/doc/p201-cash-karp.pdf