Exercise 5
Math foundation of computer graphics and vision

Amrollah Seifoddini

Task 1)

For this task, I implemented 4 functions named “analytical_diff”, “symb”, “numerical” and “automatic” for calculating derivatives. The function itself is implemented in “f_func” function. The function is f = x1^2*sin(x2) + x3/x1

By looking at the result of these functions, we see that the values for analytic and symbolic and automatic differentiations are exactly the same for all 100 random numbers. But the values for numerical methods are not precise comparing to analytical values. I observed big errors (up to 35 units) for big values of h (h=1 or h=10^-1) and as “h” gets smaller down to 10^-9 the errors are constantly decreasing and the best errors are in the range of 1^-7 for that h. If keep decreasing value of “h” to 10^-10 and 10^-11 the errors again start increasing and go up to 1^-4. That’s because in that range of h, the effect of rounding is huge for evaluation of derivatives. The formula for numerical derivative is not stable in these cases since h is in denominator and slight variation of that changes the result a lot. You can run the scrip task1.m and see the values and errors. The numbers are randomly chosen in range of [-10, 10].

Task 2)

For this task, I implemented gradient descent for all 4 derivative function. The process is repeated for 5 random points. The numbers are randomly chosen in range of [-100, 100]. The results are shown below. The pictures are also available in screenshot folder. As we can see, the energy for most of the numbers and functions has gone very fast towards –Infinity and therefore it has caused NaN for any iteration after that occurrence. That’s why those plots are discontinued after some iterations. That shows the random point is in a good place where it doesn’t stuck in local optima, and can move to –Inf. So, the minimum of energy function is –Inf. Also the step size fluctuate at the beginning but stabilize after reaching the energy of –Inf. Also we can see a lot of fluctuations and instability in some of the numerical cases (when the blue lines are like a rectangle in the plots).

The iteration and run-time of all the cases are stored in respective value cells.

For example, the runtime of analytical for each of 5 points is as follows (in seconds):

[0.03125, 0.015625, 0.03125, 0.03125, 0.03125]

The runtime for automatic differentiation is:

[0.03125,0.03125,0.046875,0.03125,0.03125]

The runtime for symbolic is (huge):

[231.4375, 252.1875,235.03125,230.875,225.078125]

And iteration numbers for all of them are:

[1000,1000,1000,1000,1000]

For numerical method I tried 6 different values for h in range of [10^0, … ,10^-10] with power step of 2. The iteration counts are like this:

H=10^0	H=10^-2	H=10^-4	H=10^-6	H=10^-8	H=10^-10
1000	51	1000	1000	24	24
340	63	1000	79	51	1000
1000	1000	1000	1000	36	168
145	1000	1000	1000	1000	44
65	56	142	98	1000	208

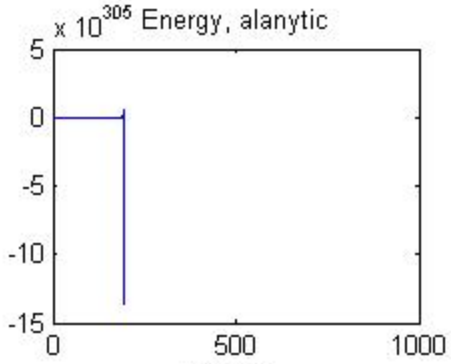
And the runtime of the same values of h are:

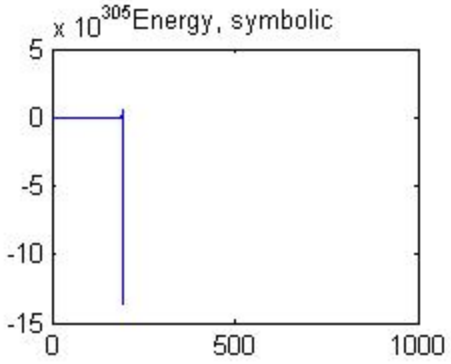
0.078125	0.015625	0.031250	0.015625	0.015625	0
0.015625	0	0.031250	0	0.015625	0.031250
0.062500	0.031250	0.031250	0.031250	0	0
0	0.046875	0.046875	0.031250	0.046875	0
0	0	0.015625	0	0.046875	0

We can see that there is no significant correlation between iteration counts and values of h, but the runtime seems to decrease for smaller h values.

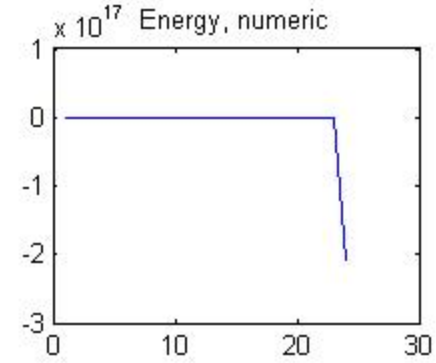
The plots of numeric method are derived with H=10^-8.

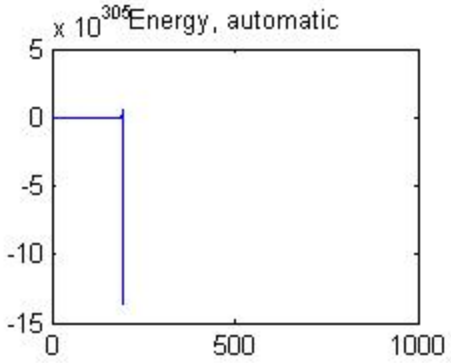
Plots for point 1:

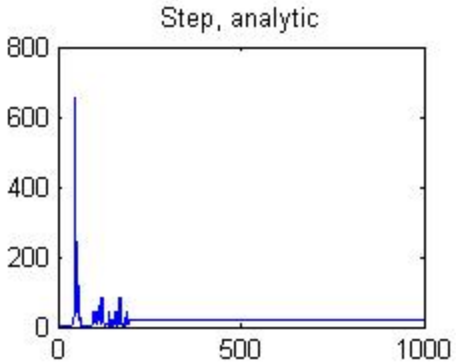

x 10^305 Energy, alanytic
5
0
-5
-10
-15
0
500
1000

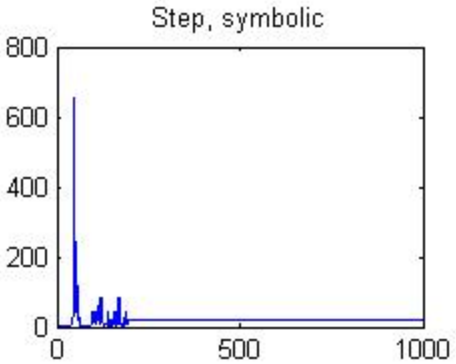

x 10^305 Energy, symbolic
5
0
-5
-10
-15
0
500
1000

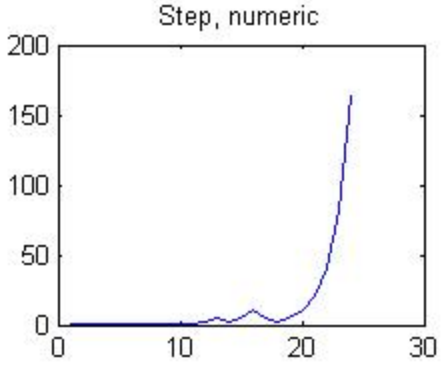
iteration

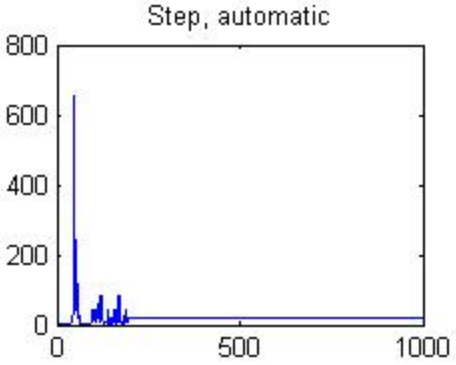

x 10^17 Energy, numeric
1
0
-1
-2
-3
0
10
20
30


x 10^305 Energy, automatic
5
0
-5
-10
-15
0
500
1000

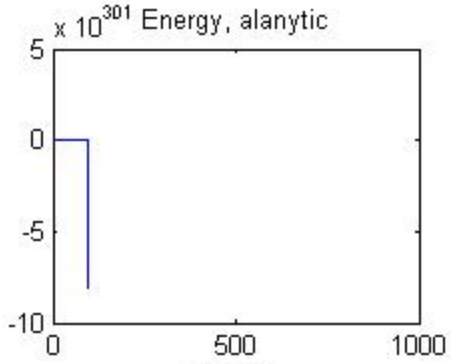

Step, analytic
800
600
400
200
0
0
500
1000

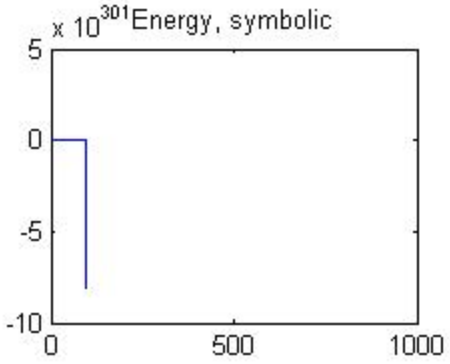

Step, symbolic
800
600
400
200
0
0
500
1000


Step, numeric
200
150
100
50
0
0
10
20
30

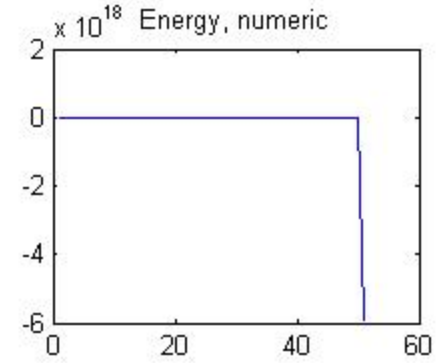

Step, automatic
800
600
400
200
0
0
500
1000

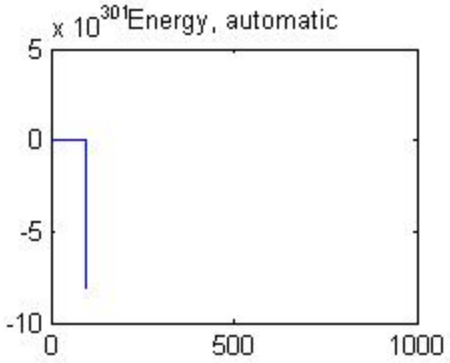
Plots for point 2:

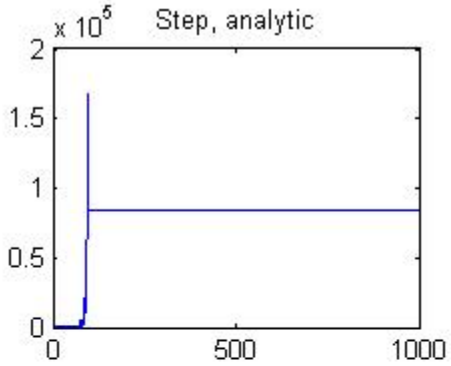

x 10^301 Energy, alanytic
5
0
-5
-10
0
500
1000

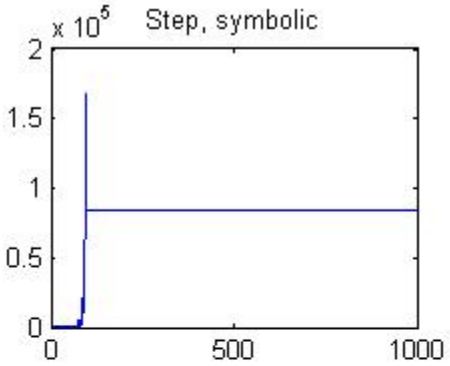

x 10^301 Energy, symbolic
5
0
-5
-10
0
500
1000

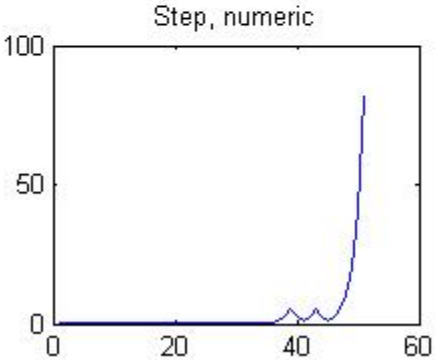
iteration

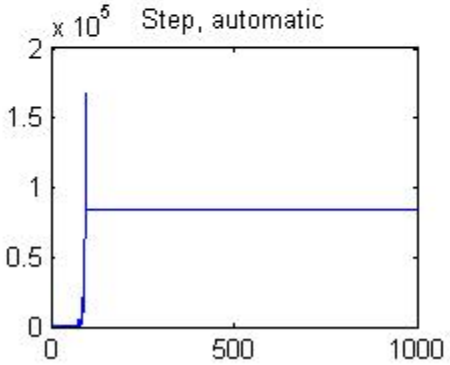

x 10^18 Energy, numeric
2
0
-2
-4
-6
0
20
40
60


x 10^301 Energy, automatic
5
0
-5
-10
0
500
1000

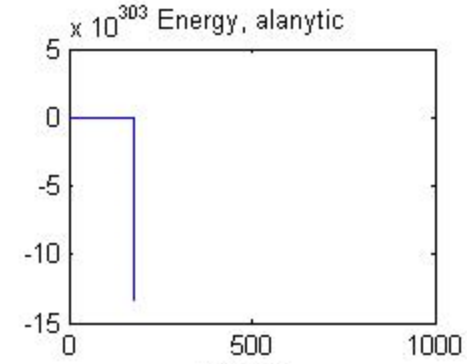

x 10^5 Step, analytic
2
1.5
1
0.5
0
0
500
1000

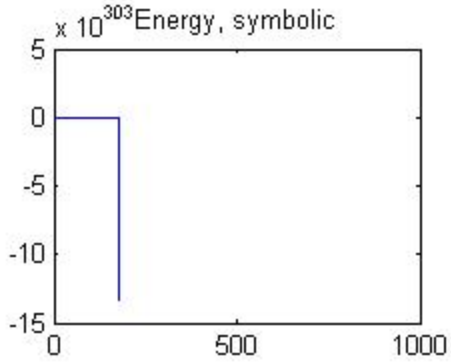

x 10^5 Step, symbolic
2
1.5
1
0.5
0
0
500
1000


Step, numeric
100
50
0
0
20
40
60

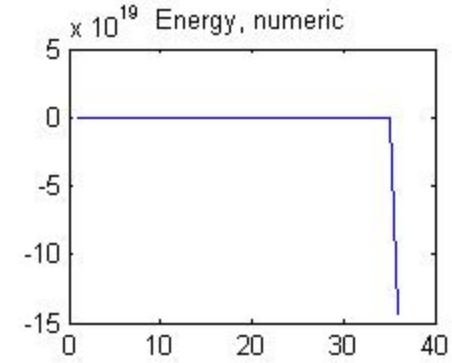

x 10^5 Step, automatic
2
1.5
1
0.5
0
0
500
1000

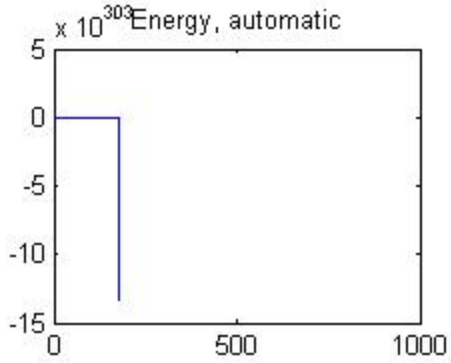
Plots for point 3:

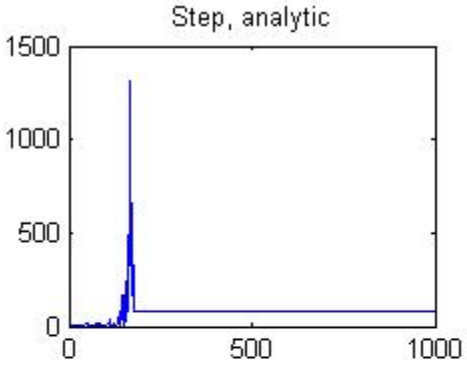

x 10^303 Energy, alanytic
5
0
-5
-10
-15
0
500
1000

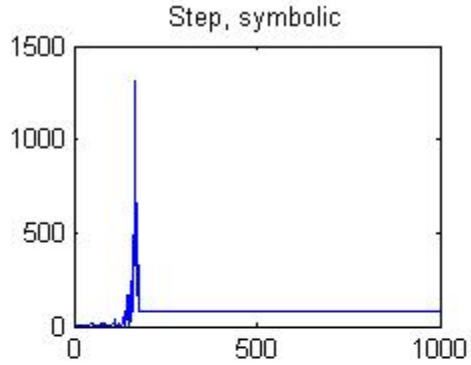

x 10^303 Energy, symbolic
5
0
-5
-10
-15
0
500
1000

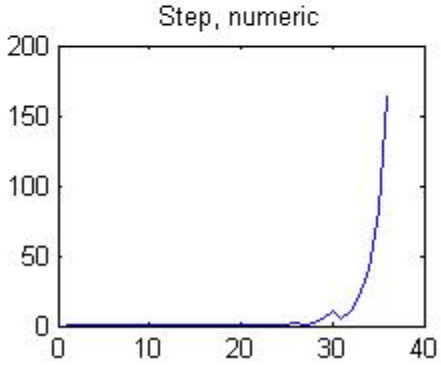
iteration

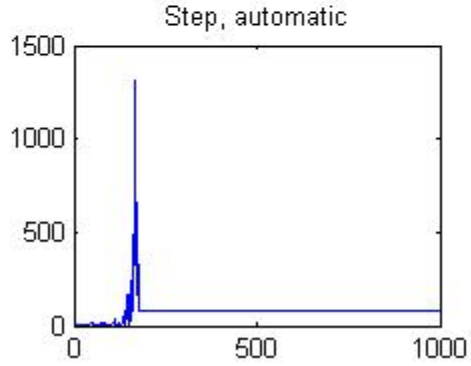

x 10^19 Energy, numeric
5
0
-5
-10
-15
0
10
20
30
40


x 10^303 Energy, automatic
5
0
-5
-10
-15
0
500
1000

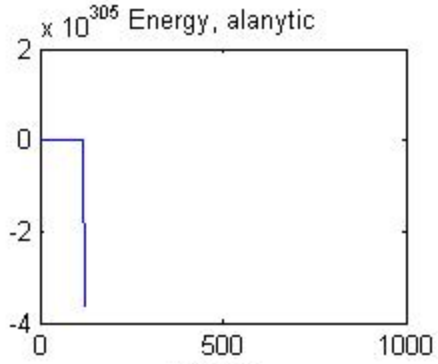

Step, analytic
1500
1000
500
0
0
500
1000

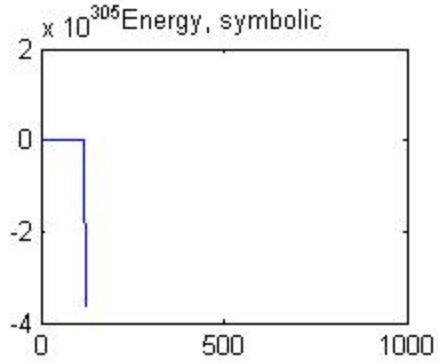

Step, symbolic
1500
1000
500
0
0
500
1000


Step, numeric
200
150
100
50
0
0
10
20
30
40

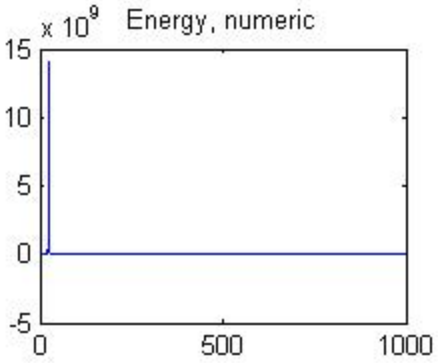

Step, automatic
1500
1000
500
0
0
500
1000

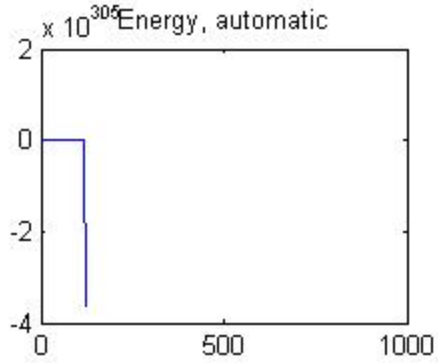
Plots for point 4:

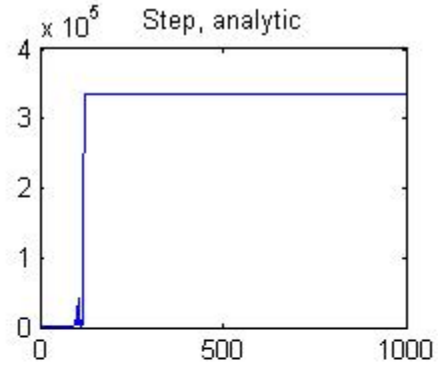

x 10^305 Energy, alanytic
2
0
-2
-4
0
500
1000

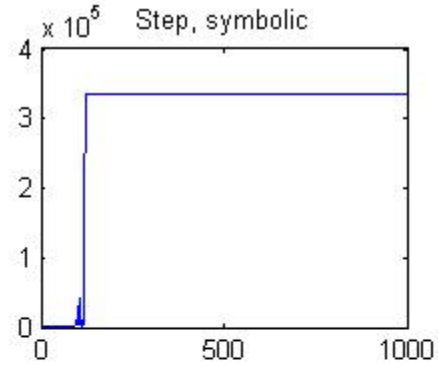

x 10^305 Energy, symbolic
2
0
-2
-4
0
500
1000

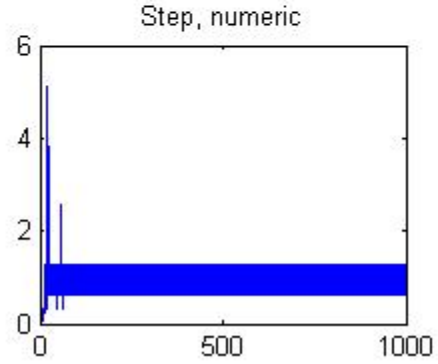
iteration

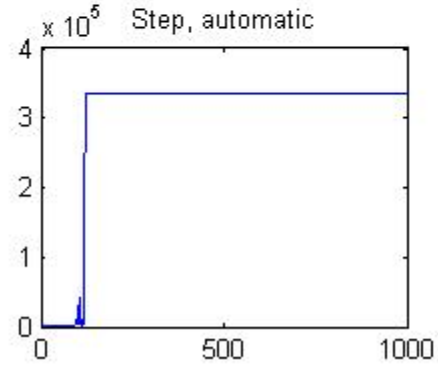

x 10^9 Energy, numeric
15
10
5
0
-5
0
500
1000


x 10^305 Energy, automatic
2
0
-2
-4
0
500
1000

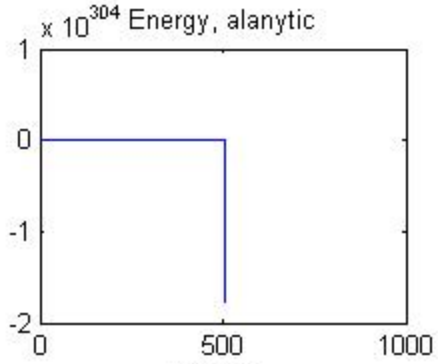

x 10^5 Step, analytic
4
3
2
1
0
0
500
1000

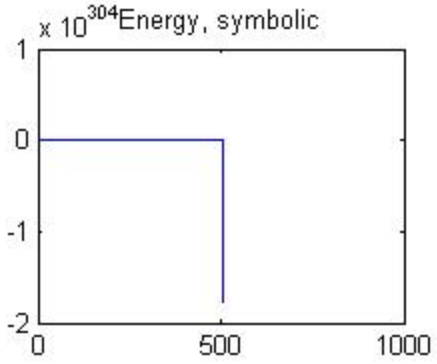

x 10^5 Step, symbolic
4
3
2
1
0
0
500
1000


Step, numeric
6
4
2
0
0
500
1000

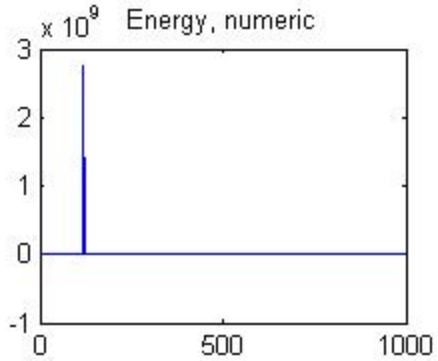

x 10^5 Step, automatic
4
3
2
1
0
0
500
1000

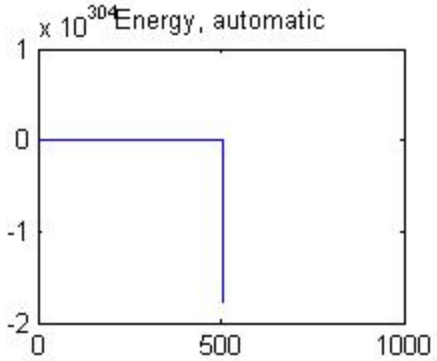
Plots for point 5:

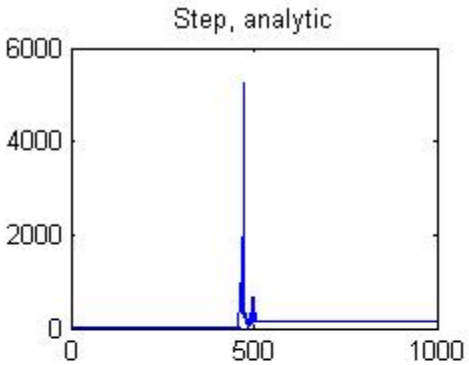

x 10^304 Energy, alanytic
1
0
-1
-2
0
500
1000

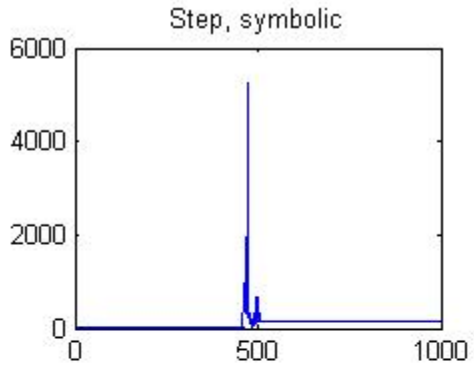

x 10^304 Energy, symbolic
1
0
-1
-2
0
500
1000

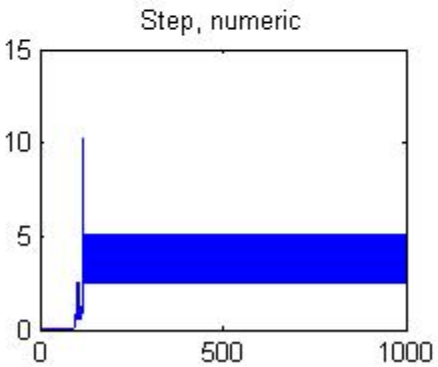
iteration

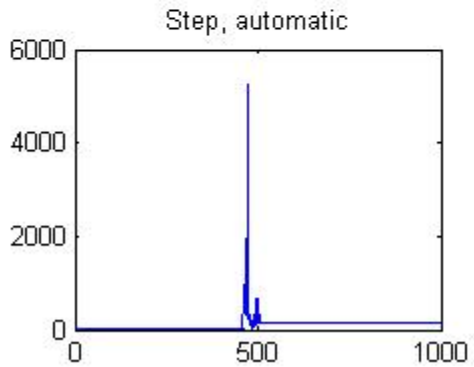

x 10^9 Energy, numeric
3
2
1
0
-1
0
500
1000


x 10^304 Energy, automatic
1
0
-1
-2
0
500
1000


Step, analytic
6000
4000
2000
0
0
500
1000


Step, symbolic
6000
4000
2000
0
0
500
1000


Step, numeric
15
10
5
0
0
500
1000


Step, automatic
6000
4000
2000
0
0
500
1000

Task 3)

The hessian matrix is formed in function hessian. Then the whole process of task2 is repeated for Newton method. The plots are displayed here. The pictures are also available in screenshot folder. In general I can say that Newton method is more stable and robust than normal gradient descent. We can confirm this by looking at pictures and see that those –Inf and NaN cases happen less here. It also reached faster to minimum value with less iterations. Therefore, the runtime is also shorter than normal gradient descent.

The iteration and run-time of all the cases are stored in respective value cells.

For example, the runtime of analytical for each of 5 points is as follows (in seconds):

[0.046875,0.031250,0.078125,0,0.078125]

The runtime for automatic differentiation is:

[0.015625,0.046875,0.281250,0.015625,0.093750]

The runtime for symbolic is (huge):

[5.781250,13.093750,18.828125,1.468750,23.171875]

And iteration numbers for all of them are:

[8,17,30,2,30]

We see obviously that number of iteration and runtime decreased drastically.

For numerical method I tried 6 different values for h in range of [10^0, … ,10^-10] with power step of 2. The iteration counts are like this:

H=10^0	H=10^-2	H=10^-4	H=10^-6	H=10^-8	H=10^-10
81	1000	1000	30	30	30
30	30	1000	1000	1000	1000
30	30	30	30	1000	30
1000	33	30	30	1000	30
31	30	1000	30	32	1000

And the runtime of the same values of h are:

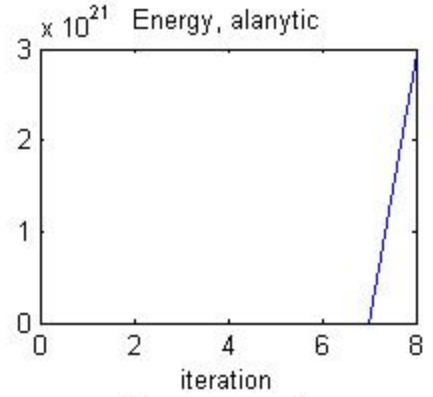
0.296875	1.76562500	1.75000	0.0312500	0.0625000	0.0156250
0.0937500	0.109375	1.15625	1.12500	1.48437500	1.17187500
0.0937500	0.0781250	0.109375	0.0625000	1.31250	0.0156250

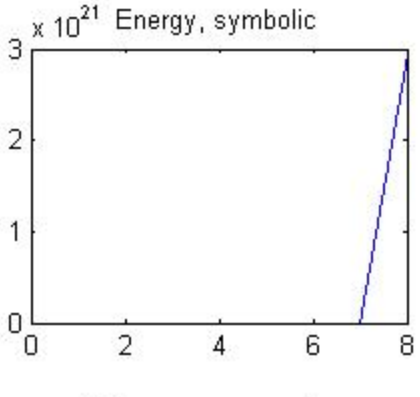
2.43750	0.109375	0.0937500	0.0781250	2.78125	0.109375
0.109375	0.140625	2.98437500	0.0937500	0.0625000	0.890625

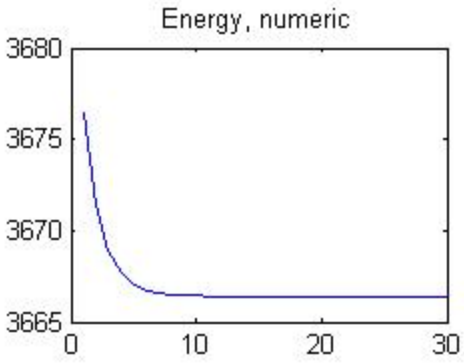
We can see that there is no significant correlation between iteration, runtime and values of h.

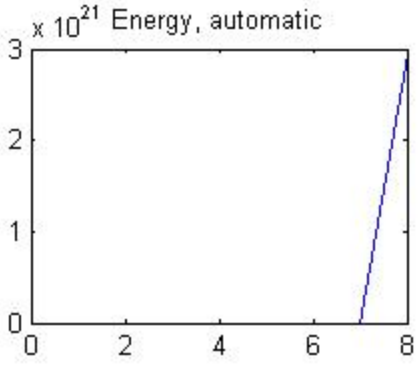
The plots of numeric method are derived with H=10^-8.

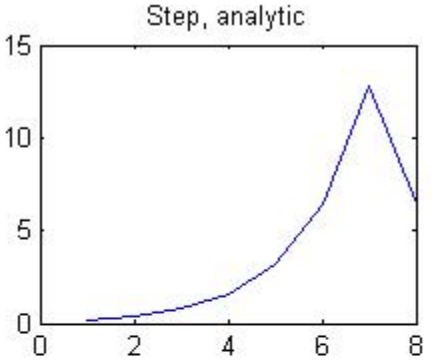
Plots for point 1:

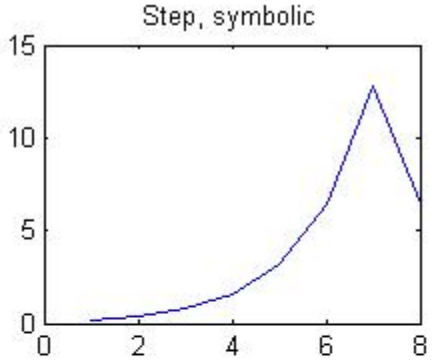

x 10^21
Energy, alanytic
3
2
1
0
0
2
4
6
8
iteration

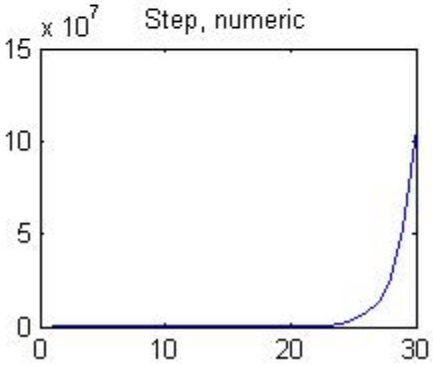

x 10^21
Energy, symbolic
3
2
1
0
0
2
4
6
8

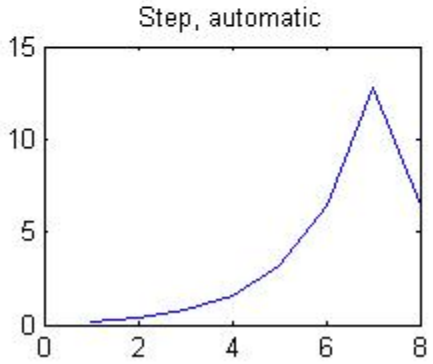

Energy, numeric
3680
3675
3670
3665
0
10
20
30


x 10^21
Energy, automatic
3
2
1
0
0
2
4
6
8

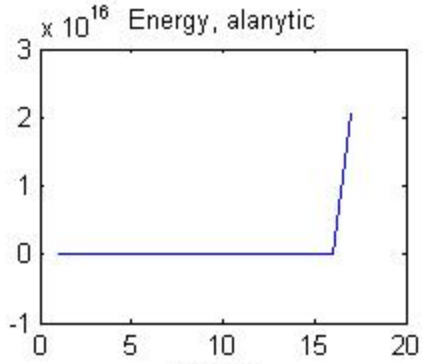

Step, analytic
15
10
5
0
0
2
4
6
8

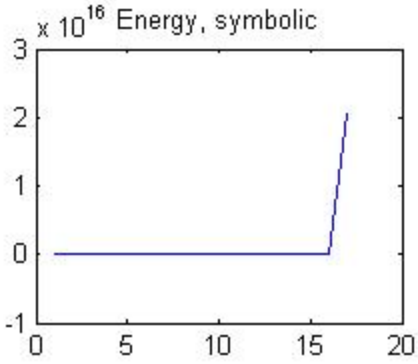

Step, symbolic
15
10
5
0
0
2
4
6
8

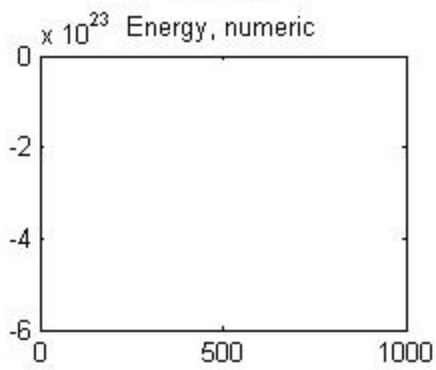

x 10^7
Step, numeric
15
10
5
0
0
10
20
30

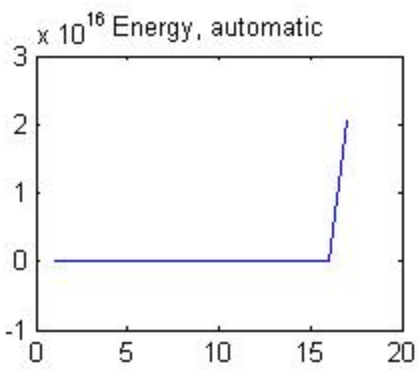

Step, automatic
15
10
5
0
0
2
4
6
8

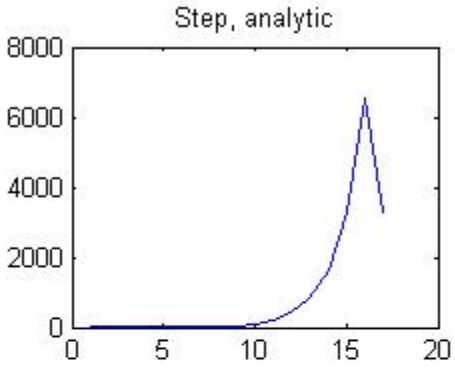
Plots for point 2:

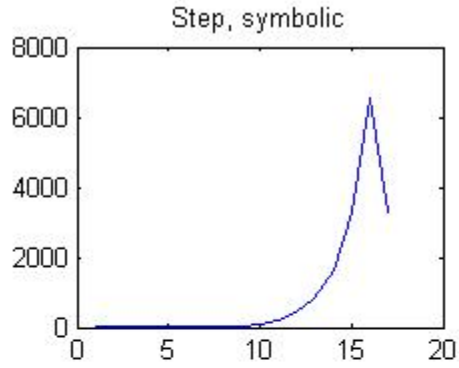

x 10^16 Energy, alanytic
3
2
1
0
-1
0
5
10
15
20

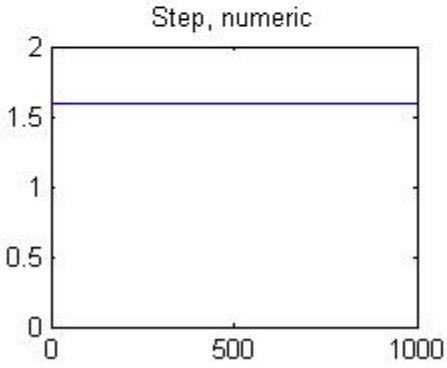

x 10^16 Energy, symbolic
3
2
1
0
-1
0
5
10
15
20

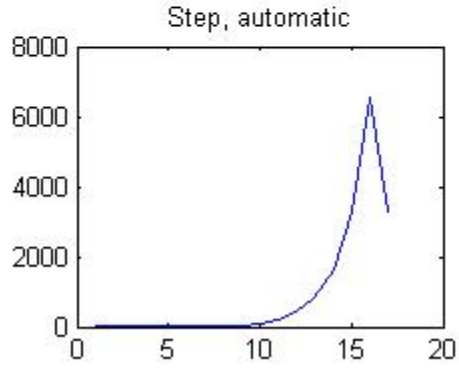

iteration
x 10^23 Energy, numeric
0
-2
-4
-6
0
500
1000


x 10^16 Energy, automatic
3
2
1
0
-1
0
5
10
15
20

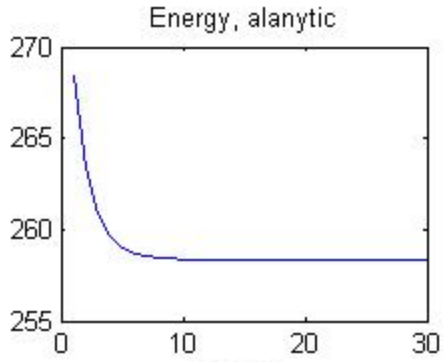

Step, analytic
8000
6000
4000
2000
0
0
5
10
15
20

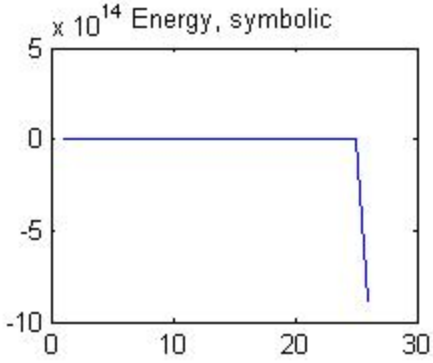

Step, symbolic
8000
6000
4000
2000
0
0
5
10
15
20

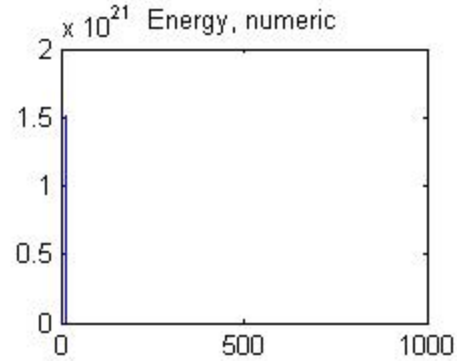

Step, numeric
2
1.5
1
0.5
0
0
500
1000

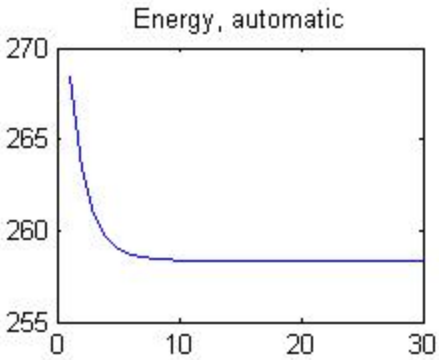

Step, automatic
8000
6000
4000
2000
0
0
5
10
15
20

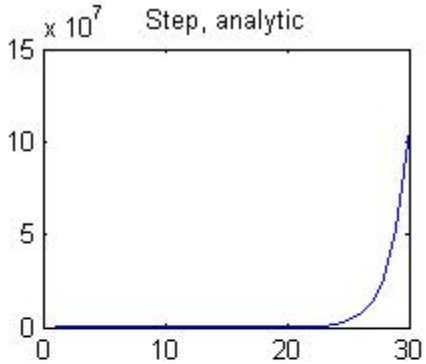
Plots for point 3:

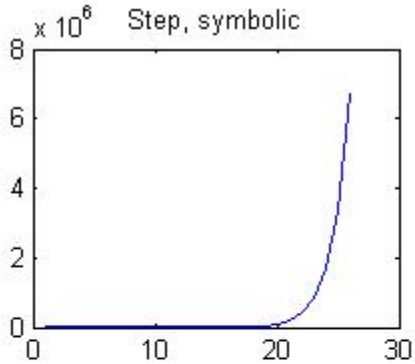

Energy, alanytic
270
265
260
255
0
10
20
30
iteration

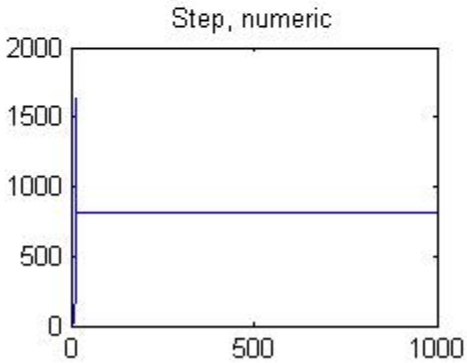

x 10^14 Energy, symbolic
5
0
-5
-10
0
10
20
30

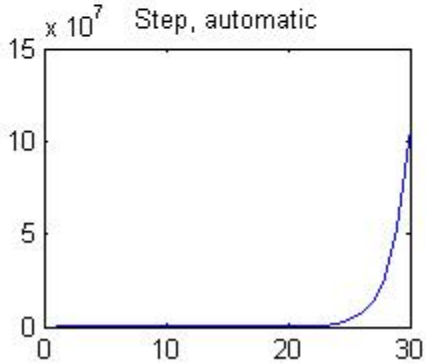

x 10^21 Energy, numeric
2
1.5
1
0.5
0
0
500
1000


Energy, automatic
270
265
260
255
0
10
20
30

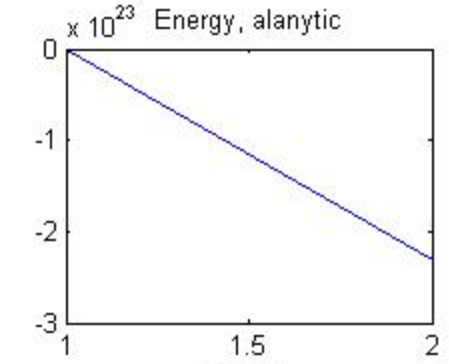

x 10^7 Step, analytic
15
10
5
0
0
10
20
30

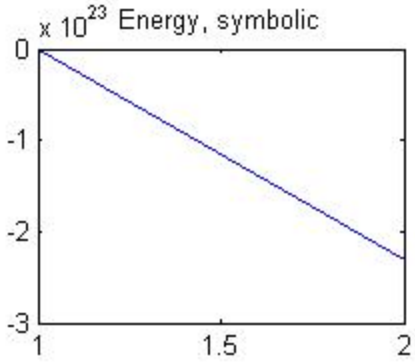

x 10^6 Step, symbolic
8
6
4
2
0
0
10
20
30


Step, numeric
2000
1500
1000
500
0
0
500
1000

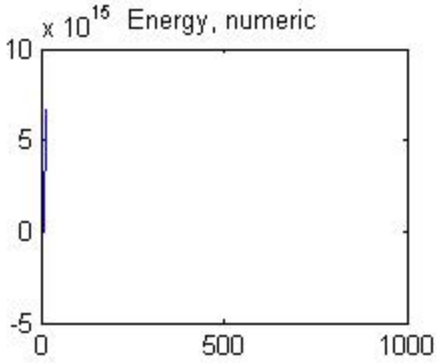

x 10^7 Step, automatic
15
10
5
0
0
10
20
30

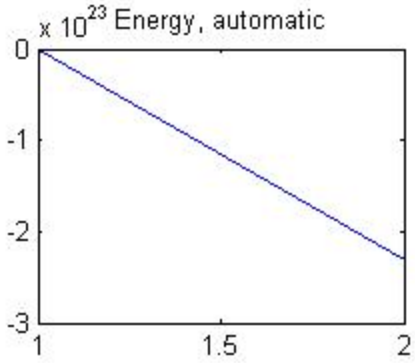
Plots for point 4:

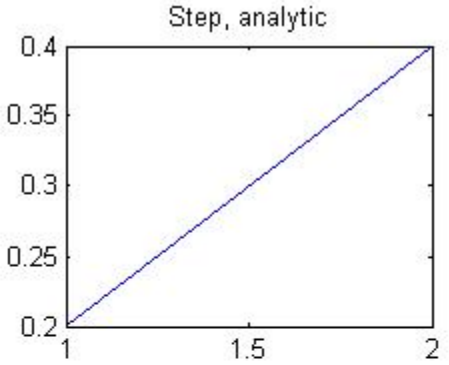

x 10^23 Energy, alanytic
0
-1
-2
-3
1
1.5
2

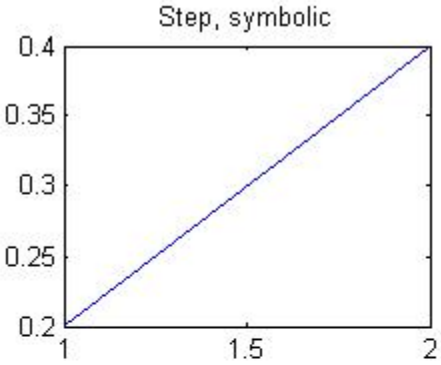

x 10^23 Energy, symbolic
0
-1
-2
-3
1
1.5
2

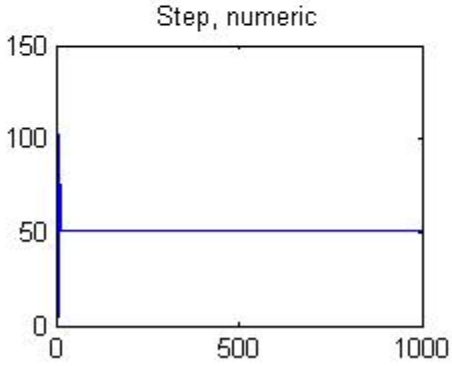
iteration

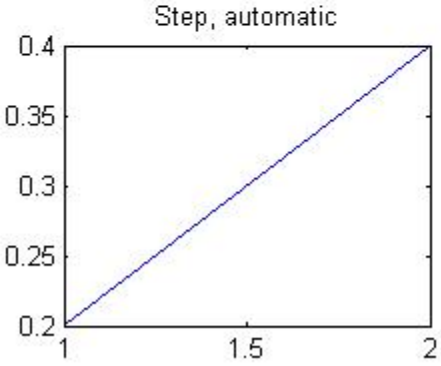

x 10^15 Energy, numeric
10
5
0
-5
0
500
1000


x 10^23 Energy, automatic
0
-1
-2
-3
1
1.5
2

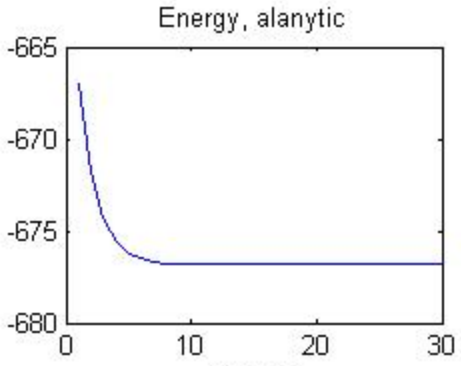

Step, analytic
0.4
0.35
0.3
0.25
0.2
1
1.5
2

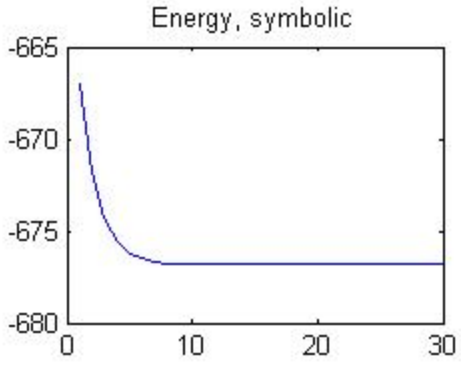

Step, symbolic
0.4
0.35
0.3
0.25
0.2
1
1.5
2


Step, numeric
150
100
50
0
0
500
1000

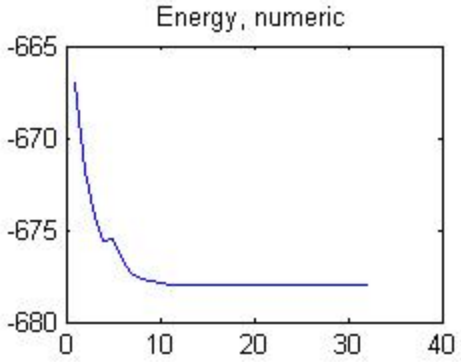

Step, automatic
0.4
0.35
0.3
0.25
0.2
1
1.5
2

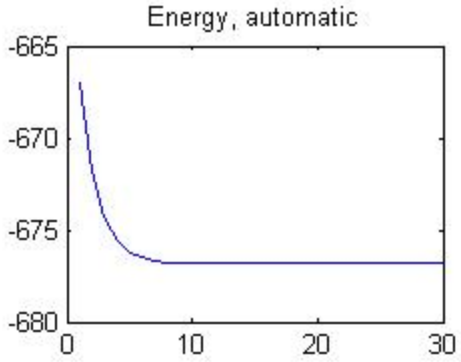
Plots for point 5:

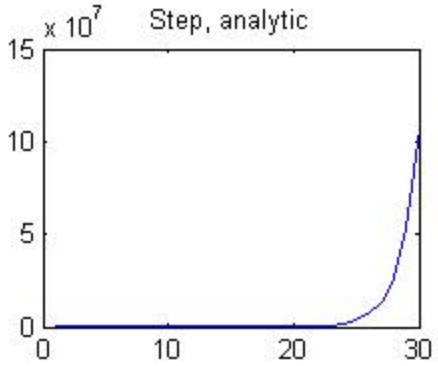

Energy, alanytic
-665
-670
-675
-680
0
10
20
30

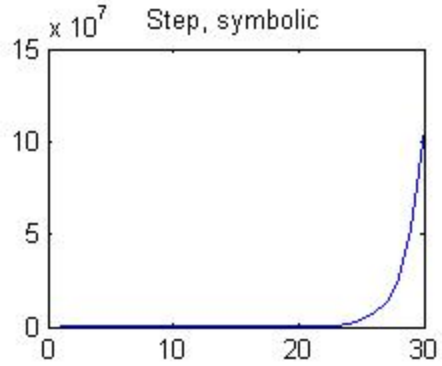

Energy, symbolic
-665
-670
-675
-680
0
10
20
30

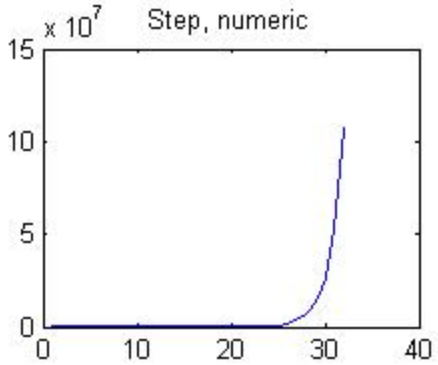
iteration

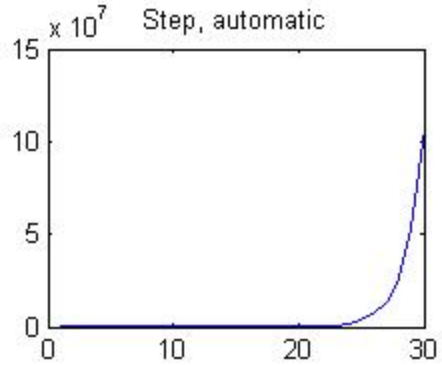

Energy, numeric
-665
-670
-675
-680
0
10
20
30
40


Energy, automatic
-665
-670
-675
-680
0
10
20
30


x 10^7
Step, analytic
15
10
5
0
0
10
20
30


x 10^7
Step, symbolic
15
10
5
0
0
10
20
30


x 10^7
Step, numeric
15
10
5
0
0
10
20
30
40


x 10^7
Step, automatic
15
10
5
0
0
10
20
30