

گزارش کار تمرین شماره ۲

ارمیا اعتمادی بروجنی

مقدمه

در این سری تمرین چند فرآکتال دیگر رسم کرده و مسئله ول نشست را بررسی می‌کنیم. کد های این سری در زبان زیبا و کاربردی Fortran نوشته شده اند و سپس نتایج مانند سری قبل به کمک برنامه Gnuplot رسم شده اند. برای کامپایل کد ها با کامپایلری مانند gfortran می‌توان به صورت زیر عمل کرد:

```
1 gfortran HW2-P1.f90 -fbackslash -o HW2-P1.out
```

البته توجه داشته باشید برای تمرین چهارم نیاز است کتابخانه جبر خطی LAPACK نیز با آرگومان مربوطه لینک شود:

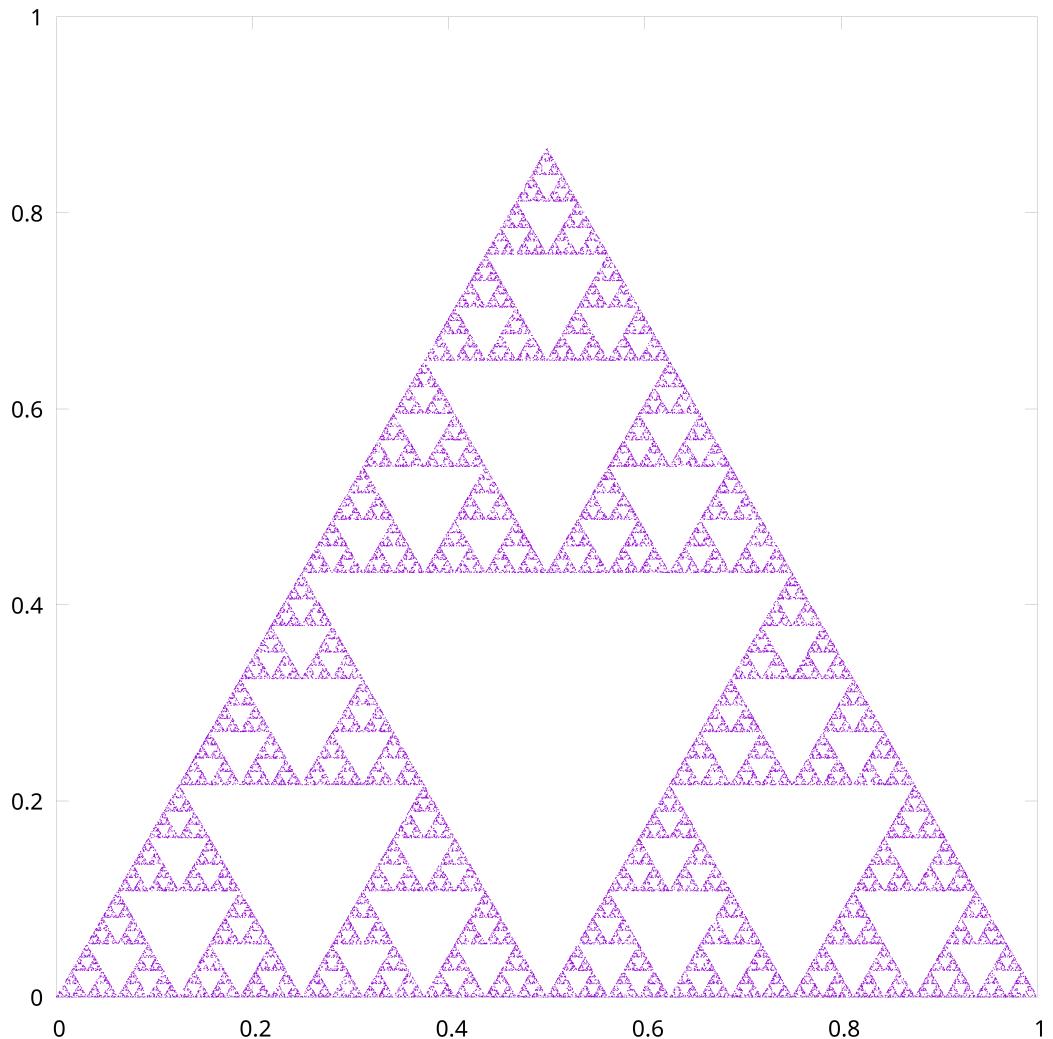
```
1 gfortran HW2-P4.f90 -fbackslash -o HW2-P4.out -L. /usr/lib/liblapack.so
```

تمرین ۱

در این تمرین به فرآکتال آشنای مثلث سرپرینسکی بازگشته و اینبار به کمک روش تصادفی آن را رسم می‌کنیم. برای این کار ابتدا تابع gen point را تعریف می‌کنیم. این تابع ابتدا به کمک ساب روتین^۱ random number در زبان فورترن، یک نقطه تصادفی با مؤلفه عمودی و افقی میان ۰ و ۱ ایجاد می‌کند. این نقطه نقطه‌ای آغازین ماست. سپس یک عدد تصادفی دیگر در همین بازه با نام choose rand ایجاد کرده که وظیفه انتخاب میان توابع مختلف مثلث سرپرینسکی را برعهده دارد. با تقسیم بندی ۰ تا ۱ به سه قسمت مساوی، تابع تصادفی انتخاب شده که در حقیقت یکی از ساب روتین های func1، func2 و func3 است. این ساب روتین ها به سادگی برای مثلث سرپرینسکی قابل نوشتن هستند. این روند برای هر نقطه اولیه ۳۰ بار تکرار می‌شود تا به یک نقطه خوب از مثلث برسیم. با این روش ۵۰۰۰ نقطه را به دست آورده و در فایل مربوطه OUT FILE ذخیره می‌کنیم. سپس با اجرای دستور gnuplot نقاط رسم می‌شوند. پarameter های رسم از قبل در فایل PLT SAVE نوشته شده اند. نتیجه به صورت زیر است:

^۱ در زبان فورترن، توابعی که بجای خروجی دادن روی آرگومان ها اثر می‌گذارند ساب روتین نامیده می‌شوند.

Sierpiński triangle



نتیجه رضایت بخشی است. همچنین من برای اطمینان از عملکرد مناسب کد، زمان های محاسبه نقاط و رسم را به صورت جداگانه بررسی کردم

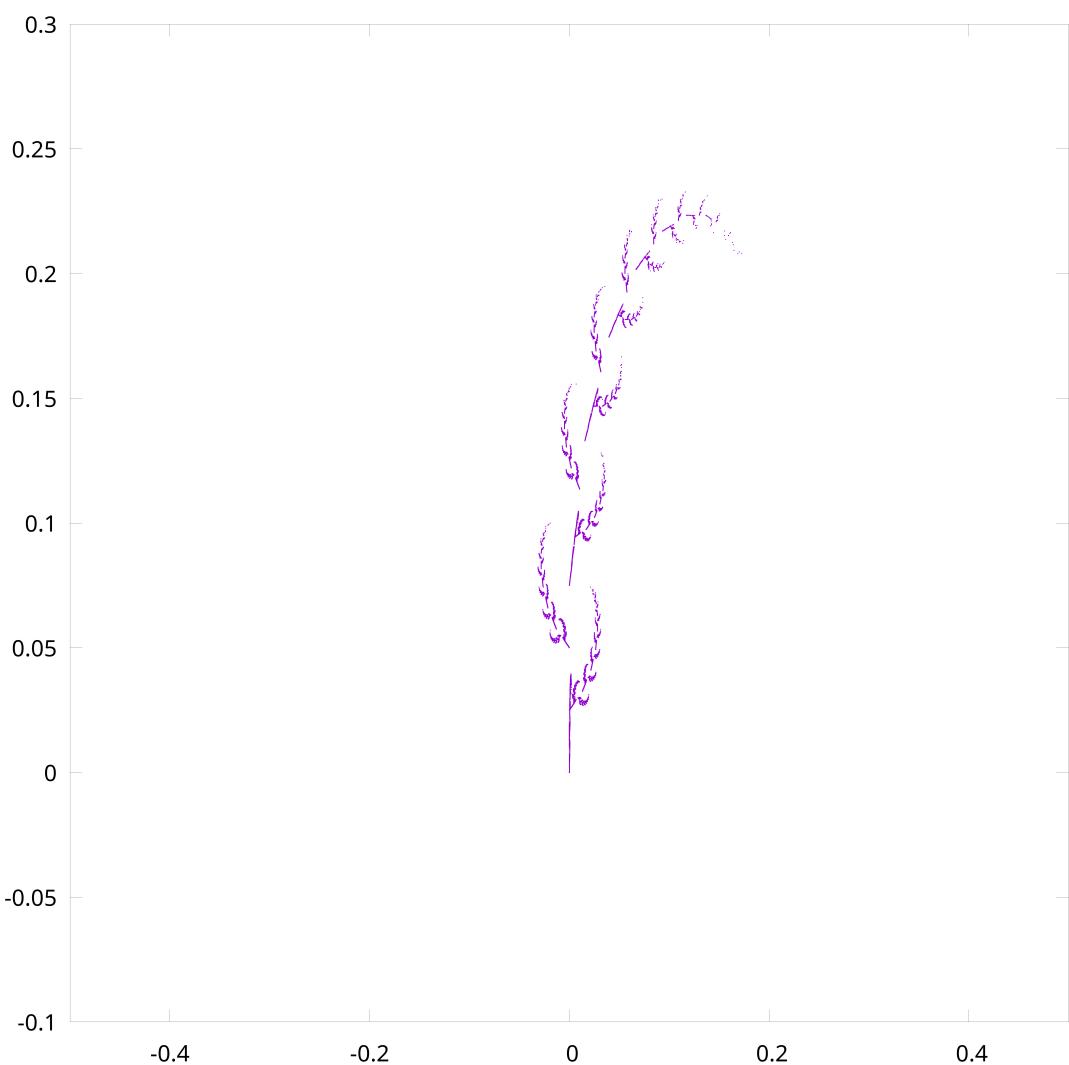
که نتیجه به صورت زیر بود:

1	Points created in:	0 seconds
2	Points plotted in:	3 seconds

داخل پرانتز: خط شامل کامند sed کار خاصی انجام نمی دهد، فقط فورترن توان را با E نمایش می دهد که لازم است در فایل خروجی برای سازگاری با گنوبلات به e تبدیل شود!

تمرین ۲

در این تمرین مشابه بخش قبل عمل می کنیم، با تفاوت که در تبدیلات دو تبدیل دوران و انعکاس اضافه شده‌اند. انعکاس نسبت به محور عمودی با قرینه کردن مؤلفه ایکس به سادگی حاصل می‌شود. برای دوران نیز پس از تعریف ماتریس دوران کافیست از تابع ضرب ماتریسی در فورترن یا استفاده کنیم. نتیجه نهایی با 100000 نقطه به صورت زیر است:



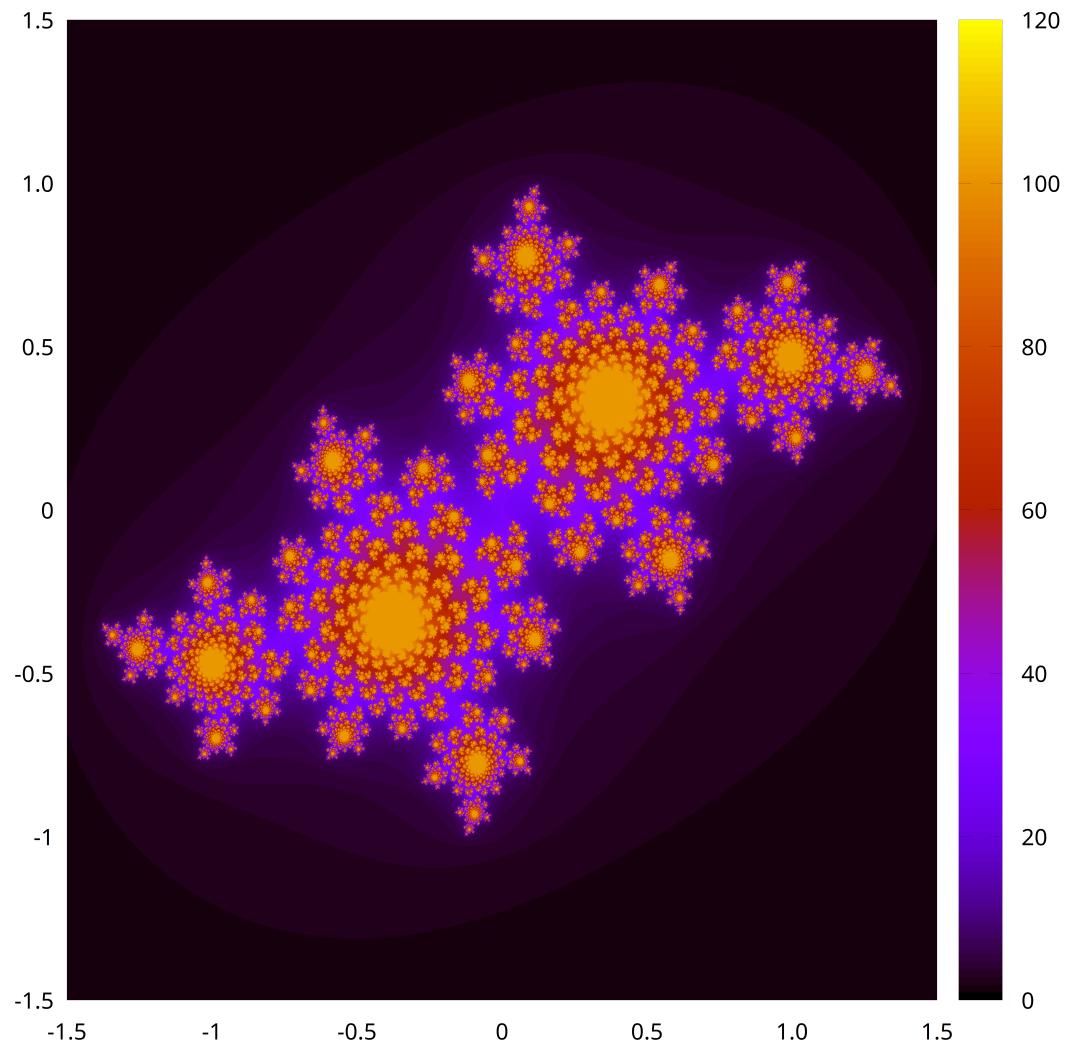
شکل برگ سرخس به خوبی نمایان است اما ناپیوستگی هایی وجود دارند که ناشی از کامل نشدن شکل (لازم داشتن نقاط بسیار بالا در روش تصادفی) و شاید بی دقتی در انتخاب پارامتر های رسم باشد. ران تایم این کد نیز به صورت زیر است:

تمرین ۳

1	Points created in:	4 seconds
2	Points plotted in:	17 seconds

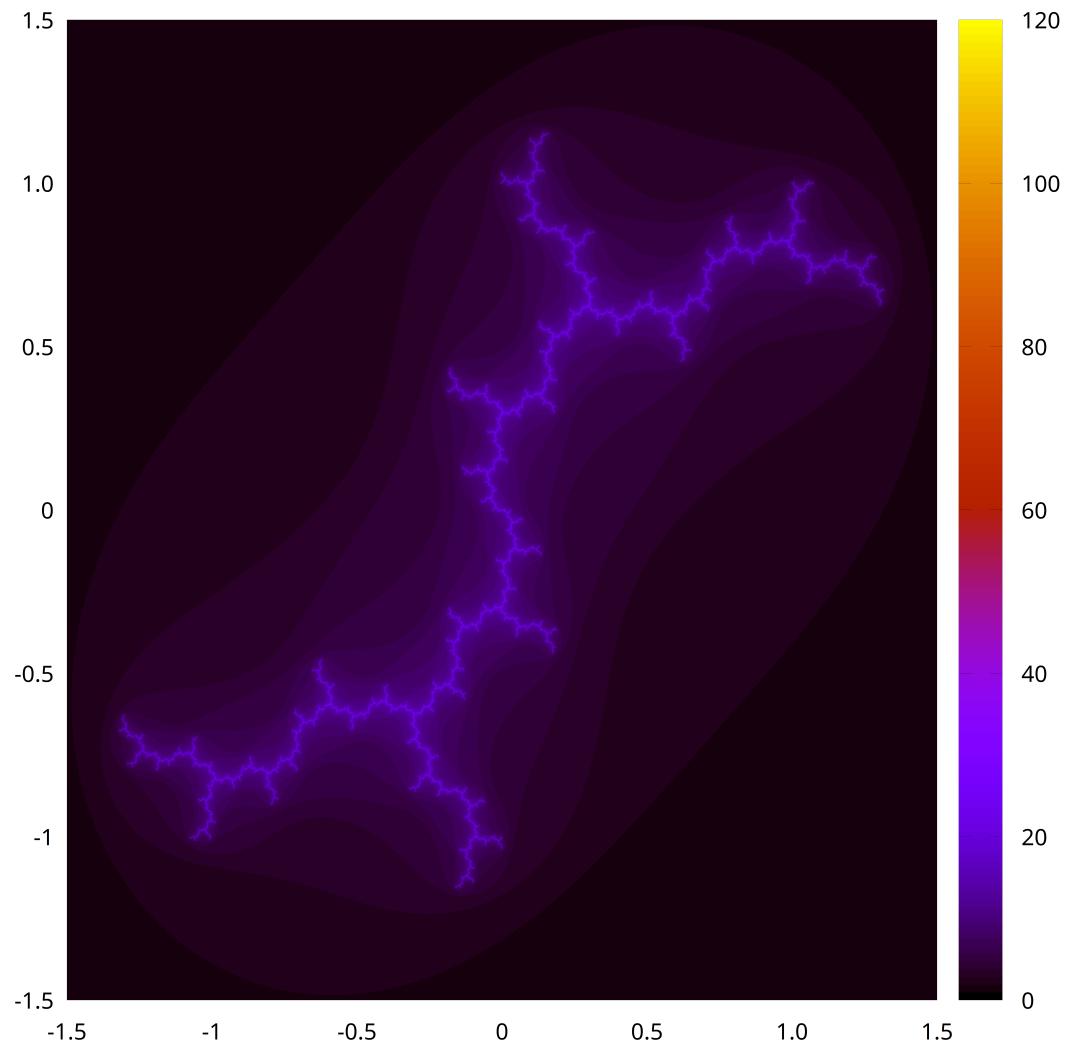
در این تمرین از روش های تصادفی فاصله گرفته و از اعداد مختلط استفاده می کنیم. زبان فورترن به صورت پیش فرض از متغیر های مختلط نیز پشتیبانی می کند. پس کافیست پارامتر C را تعریف کرده و سپس یکتابع مانند point eval تعریف کنیم که با تکرار تابع مربوطه جولیا بتواند عدد مربوط به آن نقطه را پیدا کند. در اینجا تعداد تکرار ۱۰۰ گرفته شده که هم مرز مناسبی می دهد و هم تفیکیک رنگی خوب است. سپس این باید این تابع را روی تمام صفحه مورد نظر اعمال کنیم. این کار نیز سخت نیست و با می توان نقاط صفحه را به یک ماتریس به اندازه تعداد پیکسل های صفحه مپ کرد. در اینجا کیفیت ۳۰۰۰ پیکسل در نظر گرفته شده است. همچینین نیازی نداریم تمام درایه های ماتریس همزمان در رم ذخیره شوند و در کد تنها هر سطر در یک آرایه قرار گرفته که درون فایل نوشته می شود. سپس در گنوپلات به کمک استایل های matrix image میتوان شکل های مورد نظر را رسم کرد. گرادیان رنگی پیشفرض گنوپلات نیز بنظر مناسب است. ۶ شکل کتاب در ادامه آورده شده اند:

Julia Set



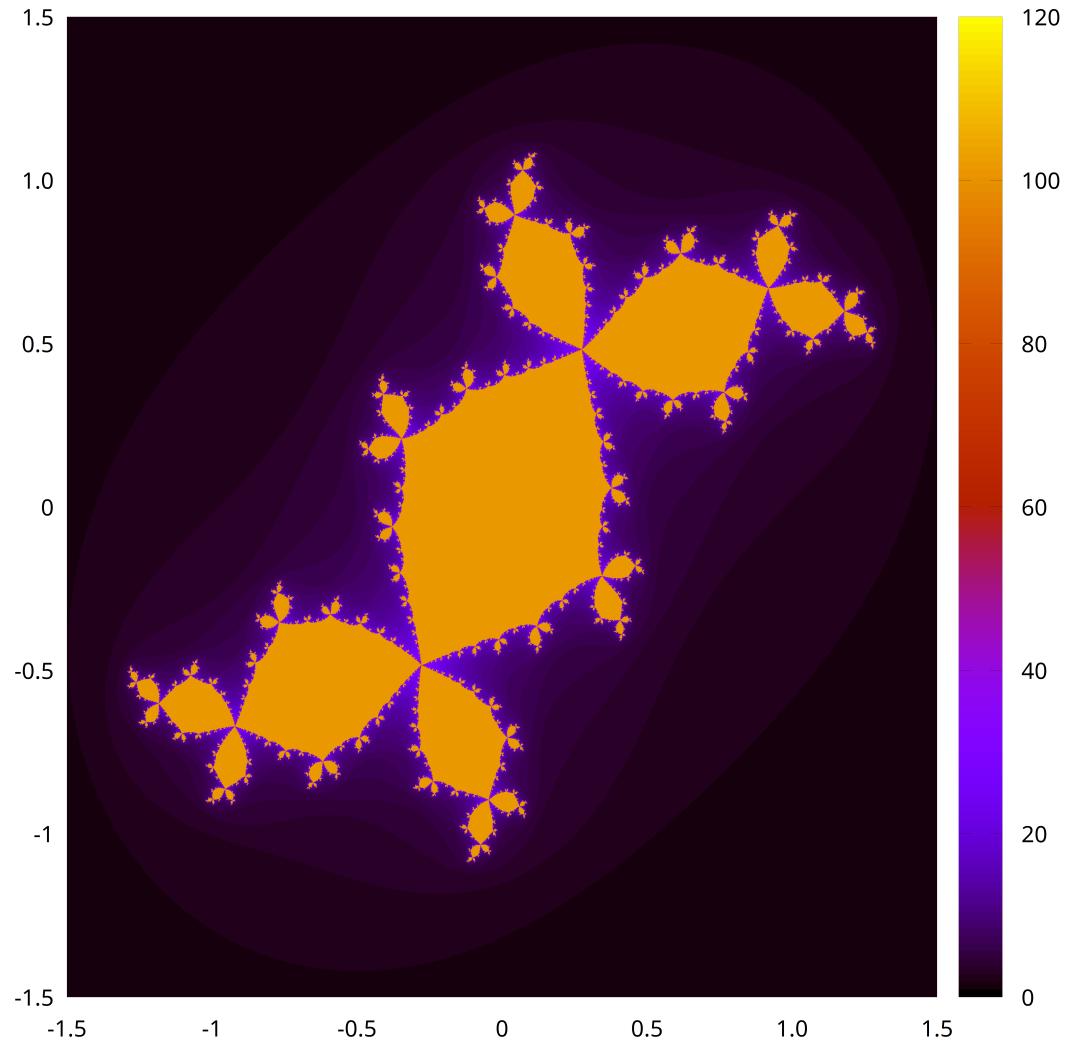
δ

Julia Set



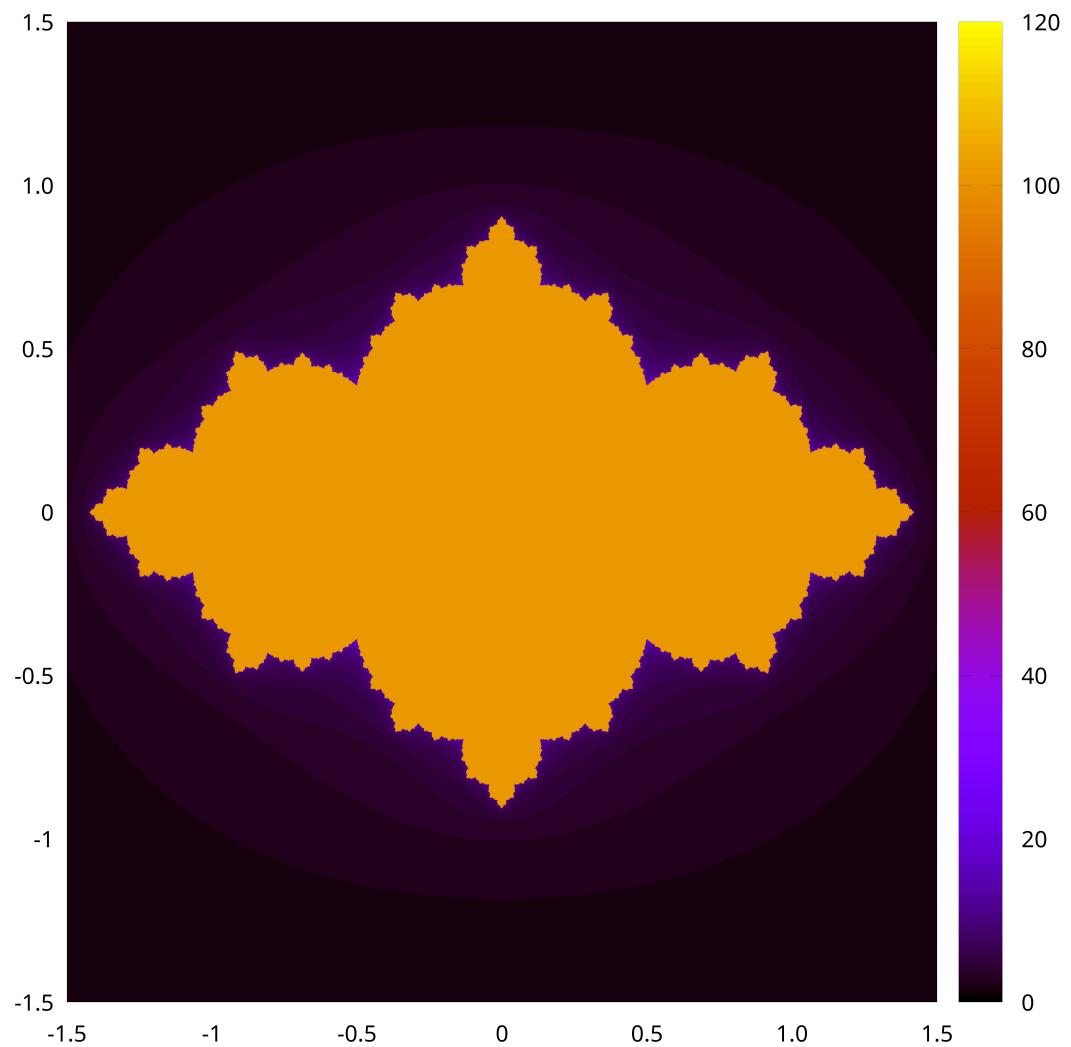
6

Julia Set



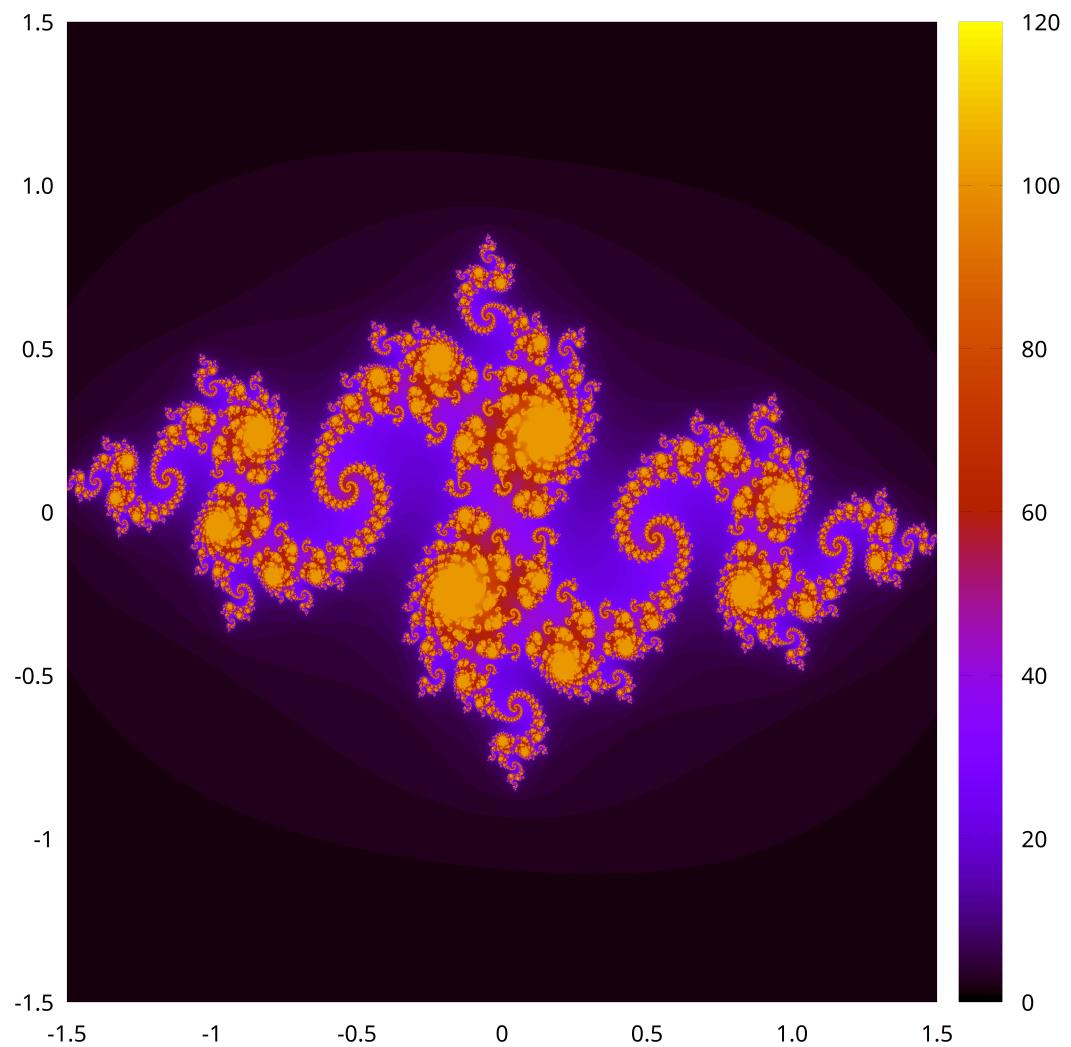
v

Julia Set

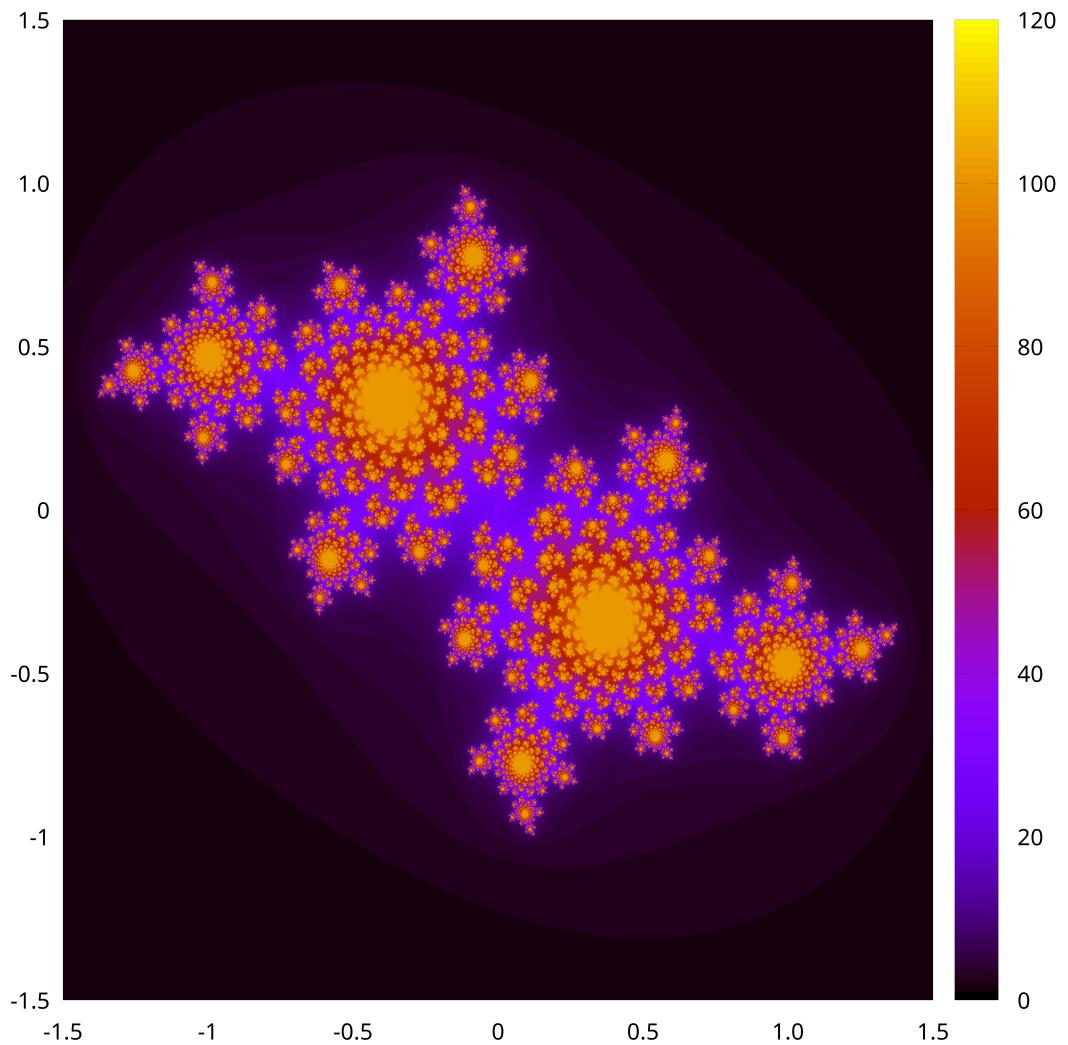


\wedge

Julia Set

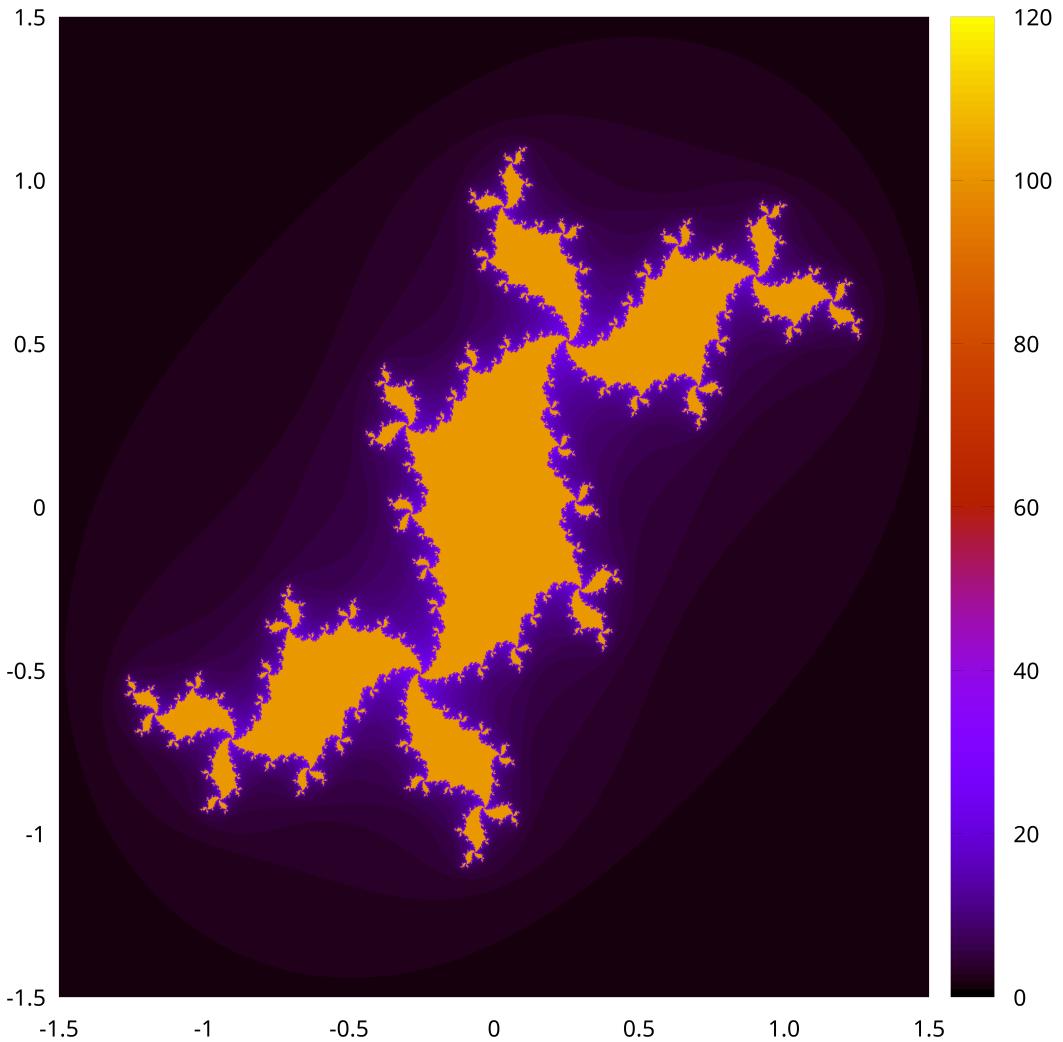


Julia Set



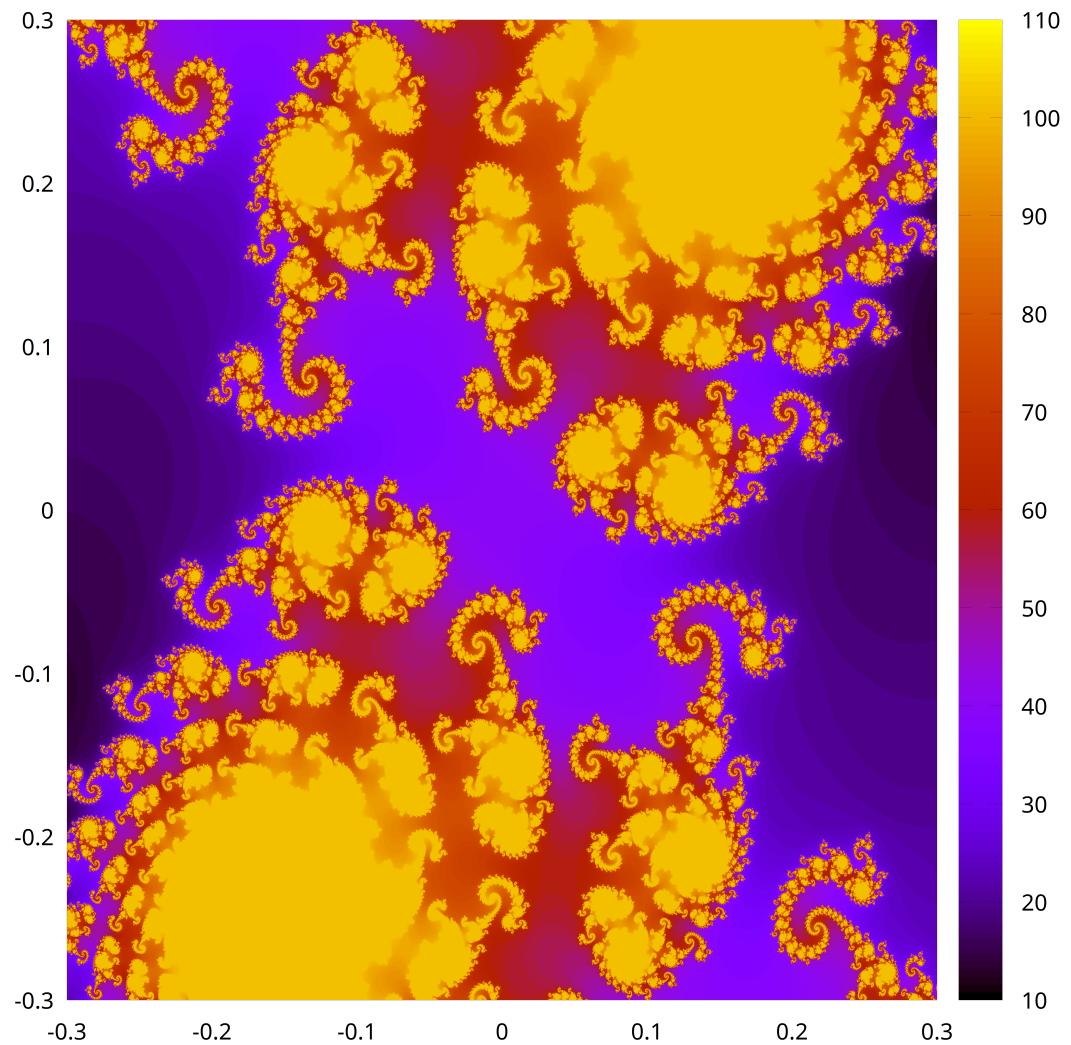
همچینیں با پارامتر دلخواه $C = -0.06 - 0.77i$ شکل زیر به دست آمد:

Julia Set



که بسیار زشت است. کیفیت تصاویر به دست آمده خوب است. با این حال با زوم کردن پیکسل ها به راحتی مشخص می شوند. با این حال می توان با تغییر مپ ماتریس به مختصات به اندازه کافی در شکل زوم کرد و آن را کاوش کرد. به عنوان مثال شکل زیر با زوم ۵ برابری در یکی از شکل های نمونه کتاب به دست آمده است:

Julia Set

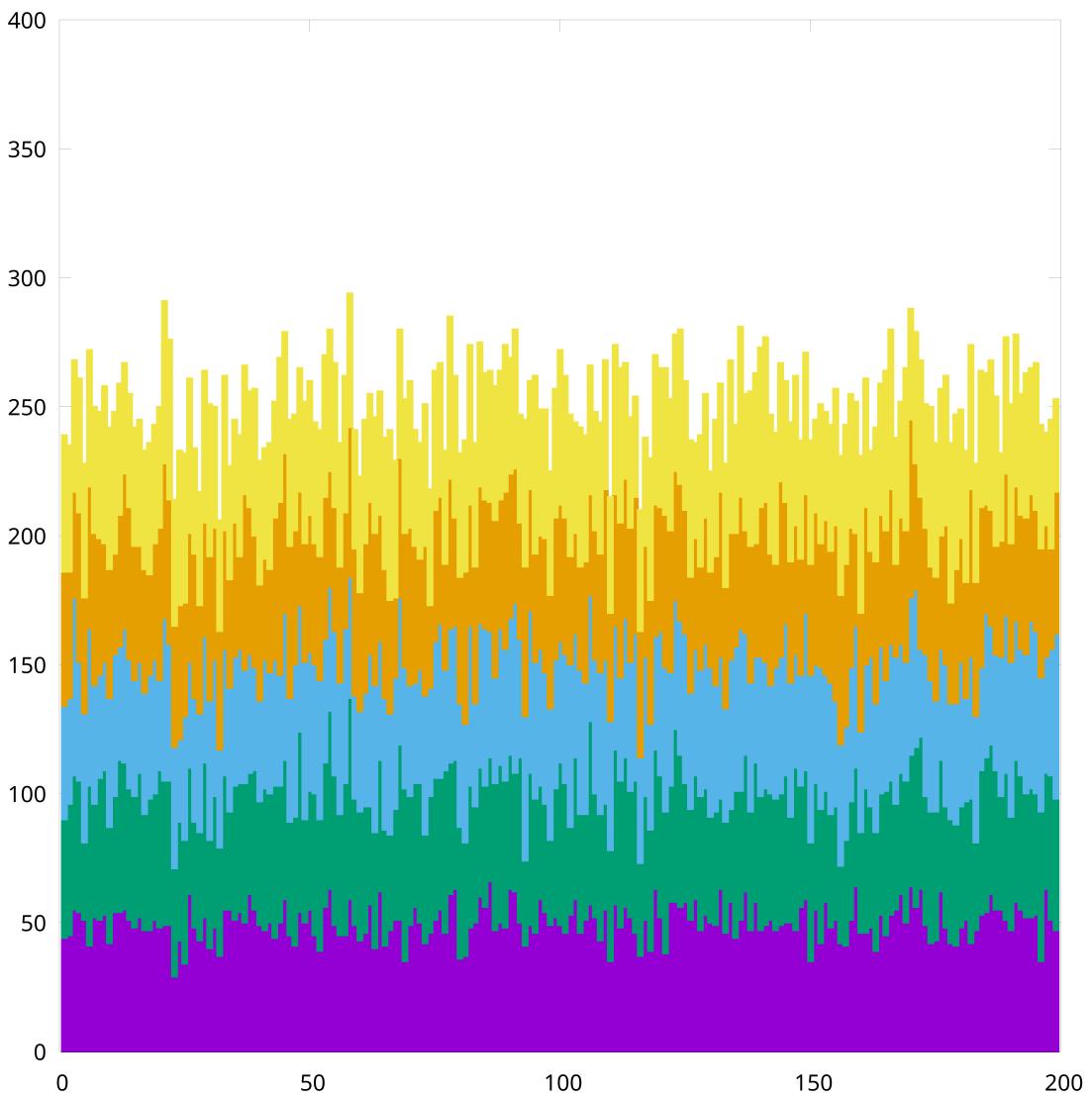


بنچمارک کد:

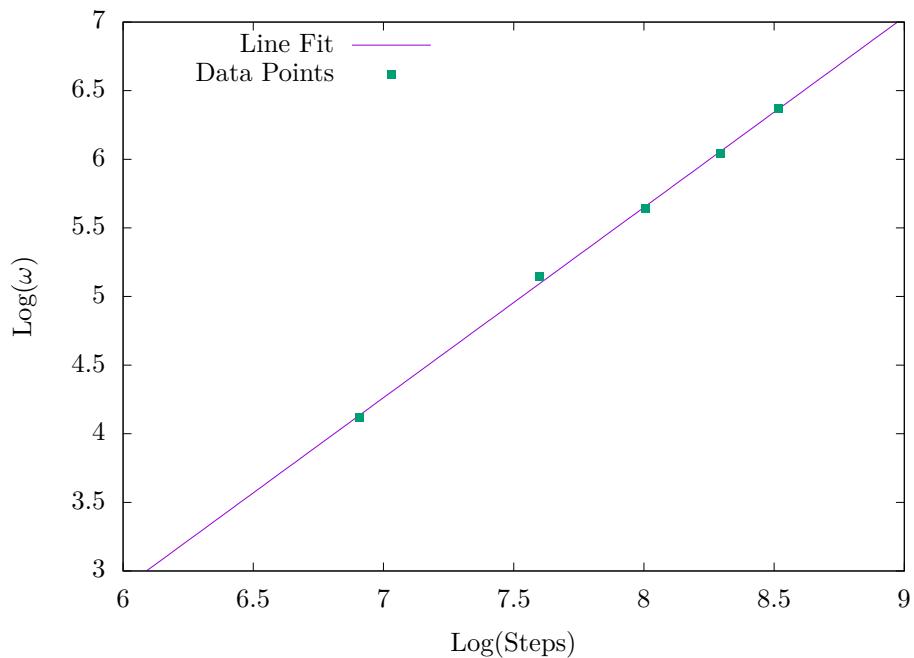
1	Points created in:	7 seconds
2	Points plotted in:	14 seconds

تمرین ۴

برای مسئله ول نشست باید در هر مرحله یک جایگاه را برای اضافه کردن ذره انتخاب کیم. برای این کار از همان ساب رو تین random number استفاده می کنیم و آن را به رنج ۱ تا L در اعداد گسسته می بریم. نتیجه این کار، تابع get rpos است. کد برای یک طول ۲۰۰ واحدی فرایند لایه نشانی را ۵۰۰۰۰ قدم انجام می دهد. همچنین در هر ۱۰۰۰۰ قدم از میانگین ارتفاع و ناهمواری نمونه برداری می شود و رنگ ذرات نیز تغییر می کنند. این نتایج در فایل های مناسب ریخته می شوند. نتیجه بصری یک لایه نشانی به صورت زیر است:



می بینیم که طبق انتظار رشد نمایی در ناهمواری (حداقل از نگاه چشم) وجود دارد. در ادامه قصد داریم در مقیاس لگاریتمی یک تابع خطی برای ناهمواری بر حسب زمان برازش کنیم. بعد از اعمال لگاریتم روی داده های ناهمواری، روش کمترین مربعات خطی مناسب کار ماست. در اینجا من از



کتابخانه LAPACK استفاده کرده ام که از شاخص ترین کتابخانه های محاسبات ماتریسی است. پس از ساختن ماتریس های مناسب برای برازش خط، این ماتریس ها به تابع calc LLS داده می شوند تا این تابع با فراخوانی روتین DGLES جواب مستله کمترین مربعات را محاسبه کند. این روتین (که یک روز از تاخیر مجاز من کم کرده است) این کار را به کمک فاکتوریزشن های مناسب از ماتریس A مانند فاکتوریزشن QR انجام می دهد. در نهایت خروجی کد در این ران به صورت زیر است:

```

1 Points created in:          0 seconds
2
3 Points plotted in:         3 seconds
4
5 Mean h:      50.0000000    100.000000    150.000000    200.000000    ←
6           250.000000
7 Var h:       61.2400017    171.520004    281.730011    420.149994    585.390015
8
9 DGELS info:             0
10 Line coefficients: -5.4461584828029741   1.3869643698103935

```

مجدد به کمک گنوپلات می توانیم داده های ناهمواری و خط برازش شده را رسم کنیم (فایل plot-P4-analyze.plt)

تابع خطی همانگونه که در شکل مشخص است مناسب است. ضریب بتا از این محاسبه چیزی حدود $\beta = 1.387$ است. برای محاسبه دقیق‌تر لازم است چندین بار لایه نشانی انجام داده و از مقادیر ناهمواری میانگین گرفت.