فاطمه سادات سیفی – 9824335

پارسا محمدپور – 98243050

سوالات تحلیلی:

سوالات کدی:

ما برای قسمت مربوط به سوالات کدی، کلا تمامی کد ها را در یک فایل قرار دادیم و یکجا آن ها را توضیح می دهیم.

برای کد این سوالات، ما به دو متغییر a و b احتیاج داریم که آنها را از ورودی می گیریم و برای این آن ها را از ورودی بگیریم، ابتدا باید آن ها را define کنیم و نوعشان را مشخص کنیم که از نوع float باشد. سپس برای خواندن یک مقدار از ورودی باید از تابع _cscanf_s() استفاده کنیم که در آن با قرار دادن %f به عنوان پارامتر اول مشخص می کنیم که نوع ورودی ای که باید دریافت کند، از نوع float است و همچنین با توجه به ورودی دومش، تعیین می شود که پس از دریافت ورودی، آن را باید در کدام متغییر بریزد (باید آدرس متغییر مورد نظر را داد) که برای همین منظور در قسمت دوم ورودی این تابع، مقدار a و یا b را با علامت & پستشاش قرار می دهیم که باعث شود مقدار ورودی را در آدرس این متغییر ها بریزد. (به دلیل وجود بحث های pointer ای و call by value و call by reference و ...).

برای قسمتی که متغییر های a و b را تعریف می کنیم و مقدارشان را از ورودی دریافت می کنیم، کد های زیر را نوشتیم:

```
float a, b; // Defining a and b as float
_cscanf_s("%f", &a); // Reading a from input
_cscanf_s("%f", &b); // Reading b from input
```

برای انجام عملیات های برداری، از آرایه ها استفاده می کنیم. همانطور که از ما خواسته شده است، برای اینکه سه بردار zi1 و zi2 و zo1 را تعریف کنیم، با توجه به اینکه بردار ها را به صورت آرایه تعریف می کنیم، پس میایم و برایشان بردار هایی به همان طول خواسته شده (50) در نظر می گیریم. خط های زیر برای تعریف کردن این بردار ها در کد قرار دادهع شده است:

```
float zi1[50], zi2[50], zo1[50]; // Defining 3 arrays for results with lentgh of 50
```

سپس باید با توجه به روابط گفته شده در مسئله، مقدار هر ایندکس این آرایه را مقدار دهی کنیم. اما برای این کار ابتدا باید ساز و کار گفته شده برای این مقدار دهی کردن را در نظر بگیریم.همانطور که در مسئله گفته شده است، ما یک y داریم که در هر گام باید به اندازه step به آن اضافه کنیم و همانطور که در مسئله گفته شده است، مقداری که در هر گام باید به آن اضافه کنیم، 0.2 می باشد. پس ابتدا y را برابر با 0 در نظر می گیریم و همانطور که از ما خواسته شده است، step را برابر با 0.2 قر ار می دهیم.(و هر سری، y را با step جمع می کنیم و در yمیریزیم و ...)
تیکه کد زیر مر بوط به این قسمتی باشد:

```
float y = 0; // Defining y
float step = 0.2; // Defining step size
```

حال باید با توجه به روابط داده شده، بردار (آرایه) های zi1 و zi2 و zo1 را پر کنیم. برای این کار ابتدا باید در یک حلقه for به اندازه size این آرایه ها پیش برویم. برای بدست آوردن سایز این آرایه ها هم از تابع sizeof() استفاده می کنیم. به طوری که این تابع، ابتدا طول کل آزایه ای که به عنوان ورودی به آن می دهیم را به ما بر میگرداند. پس ما برای اینکه

بتوانیم، طول خود آرایه را بدست بیاوریم (تعداد ایندکس های آرایه را بدست بیاوریم) باید بیایم و عددی که این تابع (وقتی که ورودی اش خود آرایه می باشد) را به ما می دهد، تقسیم بر سایز هر ایندکس آن کنیم. (که این کار را با توجه بدینگونه انجام می دهیم که مقدار موجود در اولین ایندکس آرایه را به این تابع ()sizeof پاس می دهیم با کمک * و اسم آرایه) این کار در اصل یعنی چی؟ یعنی به اما ابتدا کل فضای حافظه اختصاص داده شده به این آرایه را می دهد، اما ما میایم و این فضا را تقسیم بر مقدار فضای مورد نیاز برای هر ایندکس آرایه می کنیم. بدین شکل، تعداد اعضای آرایه بدست می آید.

برای ضرب برداری هم همانطور که گفته شد، باید آرایه ها را تمام ایندکس هایشان را پر کنیم. برای این کار باید در یک حلقه، کل آرایه را پیمایش کنیم و در هر سری، ایندکس مورد نظر را با توجه به مقدار y و مقدار های a و b مقدار دهی کنیم. برای zi1 و zi2 که این مقدار دهی کردن ها مشخص است و صرفا در فرمول گفته شده آنها را جایگزیم می کنیم.
اما برای zo1 ، ابتدا می فهمیم که مقدار آن در اصل برابر با عبارت زیر می شود:

$$\frac{1}{2(zi1 * zi2)}$$

سپس با توجه به این موضوع برای کوتاه تر کردن کار و کمتر و کوچک تر کردن عبارت، میایم و ابتدا عبارت های zi1 و zi2 را برای آن ایندکس مورد نظر در هر سری اجرای حلقه بدست می آوریم، سپس با توجه به فرمول بالا، مقدار zo1 را بدست می آوریم.
سپس در انتهای قدار دهی کردن zo1 ، (پس از مقدار دهی کردن تمامی بردار های خواسته شده در این مرحله)، مقدار y را update می کنیم، یعنی مقدار آن را بعلاوه step می کنیم.
تیکه کدزیر برای این قسمت و بخش های توضیح داده شده در رابطه با ()sizeof می باشد:

```
for(int i = 0; i < sizeof(zil)/sizeof(*zil); i++){
    zil[i] = (y * y) - a; // zil = y^2 - a
    zi2[i] = (y * y) + b; // zi2 = y^2 + b
    zol[i] = 1 / (2 * zil[i] * zi2[i]); // 1 / (2 * (y^4 + (b - a)y^2 - a * b)
    y += step; // updating y with the step
}
```

سپس برای چک کردن جواب ها، می آییم و در هر خط، مقدار ایندکس موجود در هر کدام یک از بردار ها را چاپ می کنیم. برای انجام این کار، از تابع مربوط به print کردن استفاده می کنیم، یعنی از تابع ()_cprintf استفاده می کنیم. این تابع به عنوان ورودی اول، یک String می گیرد که شامل چیز هایی است که باید چاپ کند و در ورودی های بعدی اش، مقدار هایی را میگیرد که در String آن ها را گذاشتیم چاپ کند، یعنی مثلا برای اینکه در وسط یک String، بگوییم که یک مقدار float را چاپ کن، باید در آن جایی از String که می خواهیم آن مقدار float چاپ شود، عبارت %f را قرار دهیم. با این کار، در هنگام چاپ کردن، این تابع میاید و ر وقت به این مقدار که رسید، از ورودی های بعدی این تابع، مقدار float را می خواند و آن مقدار float را چاپ می کند.
پس برای این کار هم برای اینکه بتوانیم تمام آرایه را چاپ کنیم، مشابه سری قبل، یک حلقه قرار دادیم که کل آرایه را با توجه به روش توضیح داده شده، پیمایش می کند، و سپس در هر سری، مقدار ایندکس هر کدام از بردار های zi1 و zi2 و zo1 را در آن چاپ می کنیم و در انتها هم در String ای که می خواهیم آن را چاپ کنیم، مقدار \n را قرار می دهیم که باعث می شود پس از چاپ کردن هر ایندکس از این بردار ها، در آخر کار، به خط بعدی برود.
تیکه کد زیر برای این قسمت می باشد:

```
for(int i = 0; i < sizeof(zil)/sizeof(*zil); i++){
    _cprintf("zil: %f       zi2: %f       zol: %f \n",zil[i] , zi2[i] , zol[i]); // Printing arrays
}
```

تیکه کد هایی که مربوط به سوال یک بودند، تمام شد، حال به سراغ های مربوط به سوال دو می رویم.

ابتدا برای اینکه بتوانیم بردار خروجی را در zo2 نگه داریم، آن را تعریف می کنیم. همانطور که توضیح داده شد، آن را هم یک آرایه تعریف می کنیم.
تیکه کد زیر مرزبوط به تعریف کردن این بردار (آرایه) می باشد:

```
float zo2[50]; // Defining zo2 array
```

در این قسمت از ما خواسته شده است که کد هایی به زبان اسمبلی بزنیم. برای این کار در زبان اسمبلی اگر بخوایم مقداری را از حافظه بخوانیم، باید از رجیستر های XMM استفاده کنیم، که این رجیستر ها هم تنها مشکلشان این است که نمی توانند به طور immediate مقدار بگیرند و همچنین برای استفاده از اینها، ما باید از دستورانی که استفاده می کنیم، در انتهایشان، PS قرار دهیم و نمی شود از مقدار آنها را در رجیستر های EAX و ... ریخت. اما ما با توجه به تعریف بردار zo2، نیاز داریم تا برای محاسبه اش، از عدد 2 استفاده کنیم. برای این کار، ما ایتدا به فرمول و نحوه محاسبه zo2 دقت کردیم. سپس با توجه به اینکه ما نمی توانیم مقدار immediate در رجیستر های XMM بریزیم، پس باید بیایم و مقدار ها را در جایی از حافظه ذخیره کنیم و سپس آن ها را بخوانیم. بدین شکل می توانیم این مقدار ها را در رجیستر های XMM داشته باشیم.
اما حالا برای این کار، و با توجه به نحوه تعریف zo2، می بینیم که ما می توانیم تنها با در اختیار داشتن مقدار 0.5، مقدار این بردار zo2 را با توجه به بردار های قبلی حساب کنیم. پس برای این کار، ما اابتدا در یک حلقه(با توجه به همان نحوه پیدا کردن تعداد ایندکس های یک آرایه که از قبل توضیحشان را داده ایم)، تمام ایندکس های این بردار zo2 را برابر با 0.5 می گذاریم، سپس هر وقت د ر کد اسمبلیمان، هر وقت که بخواهیم از مقدار 0.5 استفاده کنییم، مقدار هر ایندکسی از بردار zo2 را که بخواهیم آن را مقدار دهی کنیم، مقدارش را می خوانیم که برابر با 0.5 می باشد.
تیکه کد زیر برای همین است که تمام ایندکس های بردار zo2 را مقدار دهی و برابر با 0.5 قرار دهیم:

```
for(int i = 0; i < sizeof(zo2)/sizeof(*zo2); i++){
    zo2[i] =0.5; // in asm code, we divide this number (0.5) by (zi1 * zi2)
}
```

حال باید به سراغ کد اسمبلی مان برویم. برای این کار باید ابتدا عبارت asm_ را بنویسیم و سپس کد اسمبلی ا در بین دو کروشه ({}) در جلوی این عبارت بنویسیم. برای زدن این کد اسمبلی چند نوع رجیستر داریم، رجیستر های EAX و ABX و ... و یک نوع دیگر هم رجیستر های XMM0 و XMM1 و ... .

ابتدا جدول زیر را برای اینکه توضیح دهیم هر رجیستر در اصل چه کاری را انجام می دهد، رسم می کنیم:

رجیستر	کاربرد
EBX	در اصل معادل با مقدار counter است که در هر بار اجرای حلقه، مقدار آن update می شود و نشان دهنده آن است که در هر سری کدام ایندکس (ها) باید خوانده شوند و مقدارشان را در حافظه قرار دهیم
XMM0	مقدار ایندکس counter عه بردار zi1 و در یک جای دیگر هم حاصلضرب zi1 و zi2
XMM1	مقدار ایندکس counter عه بردار zi2

مقدار ایندکس counter عه برای zo2 و همچنین مقدار نهایی zo2 می باشد	XMM2

برای مقدار دهی کردن تمامی ایندکس های بردار zo2، باید بیایم و تمام ایندکس هایش را پیمایش کنیم و سپس مقدار ایندکس های متناظر بردار های ZI1 و ZI2 ا هم از بردارشان بخوانیم و سپس مقدار موجود در ZO2 را تقسیم بر حاصل ضرب مقدار های موجود در ایندکس های متناظر بردار های ZI1 و ZI2 کنیم.
پس برای این کار باید بیایم و در طی یک حلقه، تمام ایندکس های بردار zo2 را طی کنیم. برای این کار ابتدا یک متغییر counter را در نظر می گیریم و مقدار آن را برابر با صفر قرار می دهیم.

برای اینکه بتوانیم از داخل آرایه یک مقداری را بخوانیم، باید از دستور movups استفاده کنیم. پس برای اینکه مقدار آن ایندکس عه را بتوانیم بخوانیم و آن را در XMM بریزیم، از این دستور استفاده می کنیم.

در هر سری که از این دستورالعمل استفاده می کنیم، در اصل 4 تا از ایندکس های آرایه را بر می داریم و عملیات ها را باهاشون انجام می دهیم.

بقیه عملیات ها هم مانند ضرب و تقسیم هم نکته خاصی ندارند به جز اینکه دو تا رجیستر فقط می گیرند و حاصل را در همان رجیستر اول میریزد.

سپس بعد از اینکه مقدار بردار zo2 را در هر گام از حلقه ست کردیم، باید بیایم و مقدار counter را (همان رجیستر EBX) را آپدیت کنیم. برای این منظور هم با توجه به اینکه ما در هر سری با استفاده از دستور movups، ۴ تا از ایندکس ها را می خوانیم که با توجه به اینکه سایز خود float هم ۴ بایت است، پس باید 4 * 4 بایت هر سری جلوتر برویم. پس باید مقدار counter را (که همان رجیستر EBX می باشد را) در هر پایان هر سری اجرای حلقه، بعلاوه 16 کنیم.

حال باید بررسی کنیم که آیا تمامی مقادیر موجود در بردار مقدار دهی شده اند یا خیر. برای همین منظور، با توجه به توضیحات داده شده در مورد counter و گام حلقه، پس باید بررسی کنیم که آیا تعداد این ایندکس هایی که پیمایش کردیم، کل آرایه را پوشش می دهد یا خیر.
برای این کار هم میدانیم که هر سری، 4 تا 4 تا، ایندکس ها را پیمایش می کنیم، پس باید تعداد دفعات اجرا شدن حلقه مان، 4 / 50 باشد. (طبیعتا باید سقف این تعداد باشد، تا آخرین ایندکس ها هم بررسی شوند.) و همچنین در هر سری اجرای این حلقه، مقدار counter را با 16 جمع می کنیم. پس باید چک کنیم که آیا این counter ، آیا به اندازه سقف 4 / 50 با عدد 16 جمع شده است یا خیر. پس باید چی کنیم که آیا مقدار counter به مقدار 16 * (4 / 50) رسیده است یا خیر. پس باید آن را با عدد 200 مقایسه کنیم و اگر به این مقدار رسیده باشد، یعنی باید اجرای حلقه متوقف شود و دیگر نیازی نیست که به ابتدای حلقه برگردیم. اما وقتی که مقدار 200 را قرار میدادیم، اجرای کد با مشکل مواجه می شد، با آزمون و خطا های بسیار متوجه شدیم که باید عددی که counter را با آن مقایسه می کنیم، به 16 بخش پذیر باشد. برای همین به جای 200، مقدار counter را با 208 مقایسه می کنیم.

تیکه کد زیر مربوط به این تیکه کد اسمبلی می باشد:

```
_asm{
    MOV EBX, 0; // SETTING COUNTER = 0
    LOOP_START:
        MOVUPS XMM0 , OWORD PTR zi1[EBX]; // XMM0 = ZI1[EBX]     =>    XMM0 = ZI1[COUNTER]   EACH TIME IT GETS 4 INDEX OF ARRAY AND PUT IT IN XMM0
        MOVUPS XMM1 , OWORD PTR zi2[EBX]; // XMM1 = ZI2[EBX]     =>    XMM1 = ZI2[COUNTER]   EACH TIME IT GETS 4 INDEX OF ARRAY AND PUT IT IN XMM1
        MULPS XMM0, XMM1; // XMM0 = XMM0 * XMM1     => XMM = ZI1[COUNTER] * ZI2[COUNTER]
        MOVUPS XMM2, OWORD PTR zo2[EBX]; // XMM2 = ZO2[EBX]      =>    XMM2 = ZO2[COUNTER]   EACH TIME IT GETS 4 INDEX OF ARRAY AND PUT IT IN XMM2
        DIVPS XMM2, XMM0; // XMM2 = XMM2 / XMM0       =>      ZO2[COUNTER] = ZO2[COUNTER] / (ZI1[COUNTER] * ZI2[COUNTER])
        MOVUPS OWORD PTR zo2[EBX] , XMM2; // ZO2[EBX] = XMM2     =>    ZO2[COUNTER] = XMM2
        ADD EBX, 16; // UPDATING THE COUNTER. EACH TIME WE READ 4 FLOAT AND EACH FLOAT IS 4 BYTE, SO WE SHOULF ADD 4*4 BYTE TO COUNTER
        CMP EBX, 208; // AS WE EXPLAINED, WE COPMARE THE COUNTER (50 / 4) * 16 TIMES. WE SHOULD DO THOSE OPERATION FOR ALL ARRAY INDEX
        // AND IN EACH LOOP ITERATION, WE DO THIS FOR 4 ARRAY INDEX. SO WE SHOULD DO THIS CEIL(50 / 4) TIMES. IN EACH ITERATION, WE ADD THE COUNTER BY 16.
        // SO WE HAVE TO SAY STOP THE LOOP ITERATION WHEN THE COUNTER REACHED CEIL(50 / 4) * 16. WITH TOO MANY TRIES AND ERRORS, WE FOUND CMP ONLY WORKS
        // WHEN THE IMMEDIATE NUMBER IS DIVIDABLE BY 16. SO WE PUT 208 IN THAT PLACE.
        JNZ LOOP_START; // CONTINUE THE LOOP IF EBX (COUNTER) IS LESS THAN THE NUMBER WE COMPARED
}
```

سپس برای چک کردن جواب بدست آمده و درست کار کردن و یا نکردن کد اسمبلی، میایم و مقدار بردار zo1 و همچنین مقدار بردار zo1 را چاپ می کنیم(با توجه به توابع توضیح داده شده و ...) سپس اگر مقدار ایندکس های متناظر این دو بردار یکسان بود، یعنی این کد درست کار می کند و این مقدار به درستی محاسبه شده است.

تیکه کد زیر برای همین چاپ کردن می باشد:

```
for(int i = 0; i < sizeof(zo2)/sizeof(*zo2); i++){
    _cprintf("zo2: %f       zo1: %f\n", zo2[i], zo1[i]); // Printing zo2 and to see if we calculated correctly, zo2 should be equal to zo1
}
```

حال در آخرین قسمت، تمام بردار های خواسته شده را چاپ می کنیم ایندکس به ایندکس، با توجه به همان روش توضیح داده شده، و مقدار هر ایندکسشان را چک می کنیم.تیکه کد زیر برای همین قسمت می باشد:

```
for(int i = 0; i < sizeof(zi1)/sizeof(*zi1); i++){
    _cprintf("zi1: %f       zi2: %f       zo1: %f       zo2: %f \n",zi1[i] , zi2[i] , zo1[i], zo2[i]); // Printing arrays
}
```

اسکرین شات هایی که در این قسمت قرار داده شده اند، برای حالتی است که ورودی a را برابر با ی، و وردی b را نیز برابر یک قرار داده ایم.

اسکرین شات هایی از اجرا شده قسمت های مختلف کد:

قسمت اول:

```
zi1: -1.000000        zi2: 1.000000        zo1: -0.500000
zi1: -0.960000        zi2: 1.040000        zo1: -0.500801
zi1: -0.840000        zi2: 1.160000        zo1: -0.513136
zi1: -0.640000        zi2: 1.360000        zo1: -0.574449
zi1: -0.360000        zi2: 1.640000        zo1: -0.846884
zi1: 0.000000        zi2: 2.000000        zo1: 1.#INF00
zi1: 0.440000        zi2: 2.440000        zo1: 0.465723
zi1: 0.960000        zi2: 2.960000        zo1: 0.175957
zi1: 1.560000        zi2: 3.560000        zo1: 0.090032
zi1: 2.240001        zi2: 4.240001        zo1: 0.052645
zi1: 3.000001        zi2: 5.000001        zo1: 0.033333
zi1: 3.840001        zi2: 5.840001        zo1: 0.022296
zi1: 4.760002        zi2: 6.760002        zo1: 0.015539
zi1: 5.760002        zi2: 7.760002        zo1: 0.011186
zi1: 6.840003        zi2: 8.840002        zo1: 0.008269
zi1: 8.000003        zi2: 10.000003        zo1: 0.006250
zi1: 9.240004        zi2: 11.240004        zo1: 0.004814
zi1: 10.560004        zi2: 12.560004        zo1: 0.003770
zi1: 11.960005        zi2: 13.960005        zo1: 0.002995
zi1: 13.440005        zi2: 15.440005        zo1: 0.002409
zi1: 15.000004        zi2: 17.000004        zo1: 0.001961
zi1: 16.640003        zi2: 18.640003        zo1: 0.001612
zi1: 18.360001        zi2: 20.360001        zo1: 0.001338
zi1: 20.160000        zi2: 22.160000        zo1: 0.001119
zi1: 22.039997        zi2: 24.039997        zo1: 0.000944
zi1: 23.999996        zi2: 25.999996        zo1: 0.000801
zi1: 26.039993        zi2: 28.039993        zo1: 0.000685
zi1: 28.159990        zi2: 30.159990        zo1: 0.000589
zi1: 30.359989        zi2: 32.359989        zo1: 0.000509
zi1: 32.639984        zi2: 34.639984        zo1: 0.000442
zi1: 34.999985        zi2: 36.999985        zo1: 0.000386
zi1: 37.439980        zi2: 39.439980        zo1: 0.000339
zi1: 39.959976        zi2: 41.959976        zo1: 0.000298
zi1: 42.559975        zi2: 44.559975        zo1: 0.000264
zi1: 45.239971        zi2: 47.239971        zo1: 0.000234
zi1: 47.999966        zi2: 49.999966        zo1: 0.000208
zi1: 50.839962        zi2: 52.839962        zo1: 0.000186
zi1: 53.759960        zi2: 55.759960        zo1: 0.000167
zi1: 56.759956        zi2: 58.759956        zo1: 0.000150
zi1: 59.839951        zi2: 61.839951        zo1: 0.000135
zi1: 62.999947        zi2: 64.999947        zo1: 0.000122
zi1: 66.239952        zi2: 68.239952        zo1: 0.000111
zi1: 69.559944        zi2: 71.559944        zo1: 0.000100
zi1: 72.959938        zi2: 74.959938        zo1: 0.000091
zi1: 76.439934        zi2: 78.439934        zo1: 0.000083
zi1: 79.999931        zi2: 81.999931        zo1: 0.000076
zi1: 83.639923        zi2: 85.639923        zo1: 0.000070
zi1: 87.359924        zi2: 89.359924        zo1: 0.000064
zi1: 91.159912        zi2: 93.159912        zo1: 0.000059
zi1: 95.039909        zi2: 97.039909        zo1: 0.000054
Press any key to continue . . .
```

اسکرین شات از اجرا شدن قسمت دوم کد(بدون zo1 برای چک کردن):

```
zo2: 0.000000
zo2: 0.000000
zo2: 0.000000
zo2: 0.000000
zo2: 0.000000
zo2: 0.000000
zo2: 0.000000
zo2: 0.000000
zo2: 0.000000
zo2: 0.000000
zo2: 0.000000
zo2: 0.000000
zo2: 0.000000
zo2: 0.000000
zo2: 0.000000
zo2: 0.000000
zo2: 0.000000
zo2: 0.000000
zo2: 0.000000
zo2: 0.000000
zo2: 0.000000
zo2: 0.000000
zo2: 0.000000
zo2: 0.000000
zo2: 0.000000
zo2: 0.000000
zo2: 0.000000
zo2: 0.000000
zo2: 0.000000
zo2: 0.000000
zo2: 0.000000
zo2: 0.000000
zo2: 0.000000
zo2: 0.000000
zo2: 0.000000
zo2: 0.000000
zo2: 0.000000
zo2: 0.000000
zo2: 0.000000
zo2: 0.000000
zo2: 0.000000
zo2: 0.000000
zo2: 0.000000
zo2: 0.000000
zo2: 0.000000
zo2: 0.000000
zo2: 0.000000
zo2: 0.000000
zo2: 0.000000
zo2: 0.000000
Press any key t
```

اسکرین شات قسمت دوم کد (با وجود بردار zo1 برای چک کردن):

```
zo2: -0.574449        zo1: -0.574449
zo2: -0.846883        zo1: -0.846884
zo2: 1.#INF00        zo1: 1.#INF00
zo2: 0.465723        zo1: 0.465723
zo2: 0.175957        zo1: 0.175957
zo2: 0.090032        zo1: 0.090032
zo2: 0.052645        zo1: 0.052645
zo2: 0.033333        zo1: 0.033333
zo2: 0.022296        zo1: 0.022296
zo2: 0.015539        zo1: 0.015539
zo2: 0.011186        zo1: 0.011186
zo2: 0.008269        zo1: 0.008269
zo2: 0.006250        zo1: 0.006250
zo2: 0.004814        zo1: 0.004814
zo2: 0.003770        zo1: 0.003770
zo2: 0.002995        zo1: 0.002995
zo2: 0.002409        zo1: 0.002409
zo2: 0.001961        zo1: 0.001961
zo2: 0.001612        zo1: 0.001612
zo2: 0.001338        zo1: 0.001338
zo2: 0.001119        zo1: 0.001119
zo2: 0.000944        zo1: 0.000944
zo2: 0.000801        zo1: 0.000801
zo2: 0.000685        zo1: 0.000685
zo2: 0.000589        zo1: 0.000589
zo2: 0.000509        zo1: 0.000509
zo2: 0.000442        zo1: 0.000442
zo2: 0.000386        zo1: 0.000386
zo2: 0.000339        zo1: 0.000339
zo2: 0.000298        zo1: 0.000298
zo2: 0.000264        zo1: 0.000264
zo2: 0.000234        zo1: 0.000234
zo2: 0.000208        zo1: 0.000208
zo2: 0.000186        zo1: 0.000186
zo2: 0.000167        zo1: 0.000167
zo2: 0.000150        zo1: 0.000150
zo2: 0.000135        zo1: 0.000135
zo2: 0.000122        zo1: 0.000122
zo2: 0.000111        zo1: 0.000111
zo2: 0.000100        zo1: 0.000100
zo2: 0.000091        zo1: 0.000091
zo2: 0.000083        zo1: 0.000083
zo2: 0.000076        zo1: 0.000076
zo2: 0.000070        zo1: 0.000070
zo2: 0.000064        zo1: 0.000064
zo2: 0.000059        zo1: 0.000059
zo2: 0.000054        zo1: 0.000054
Press any key to continue . . .
```

اسکرین شات از قسمت سوم کد:

```
zi1: -1.000000       zi2: 1.000000       zo1: -0.500000       zo2: -0.500000
zi1: -0.960000       zi2: 1.040000       zo1: -0.500801       zo2: -0.500801
zi1: -0.840000       zi2: 1.160000       zo1: -0.513136       zo2: -0.513136
zi1: -0.640000       zi2: 1.360000       zo1: -0.574449       zo2: -0.574449
zi1: -0.360000       zi2: 1.640000       zo1: -0.846884       zo2: -0.846883
zi1: 0.000000       zi2: 2.000000       zo1: 1.#INF00       zo2: 1.#INF00
zi1: 0.440000       zi2: 2.440000       zo1: 0.465723       zo2: 0.465723
zi1: 0.960000       zi2: 2.960000       zo1: 0.175957       zo2: 0.175957
zi1: 1.560000       zi2: 3.560000       zo1: 0.090032       zo2: 0.090032
zi1: 2.240001       zi2: 4.240001       zo1: 0.052645       zo2: 0.052645
zi1: 3.000001       zi2: 5.000001       zo1: 0.033333       zo2: 0.033333
zi1: 3.840001       zi2: 5.840001       zo1: 0.022296       zo2: 0.022296
zi1: 4.760002       zi2: 6.760002       zo1: 0.015539       zo2: 0.015539
zi1: 5.760002       zi2: 7.760002       zo1: 0.011186       zo2: 0.011186
zi1: 6.840003       zi2: 8.840002       zo1: 0.008269       zo2: 0.008269
zi1: 8.000003       zi2: 10.000003       zo1: 0.006250       zo2: 0.006250
zi1: 9.240004       zi2: 11.240004       zo1: 0.004814       zo2: 0.004814
zi1: 10.560004       zi2: 12.560004       zo1: 0.003770       zo2: 0.003770
zi1: 11.960005       zi2: 13.960005       zo1: 0.002995       zo2: 0.002995
zi1: 13.440005       zi2: 15.440005       zo1: 0.002409       zo2: 0.002409
zi1: 15.000004       zi2: 17.000004       zo1: 0.001961       zo2: 0.001961
zi1: 16.640003       zi2: 18.640003       zo1: 0.001612       zo2: 0.001612
zi1: 18.360001       zi2: 20.360001       zo1: 0.001338       zo2: 0.001338
zi1: 20.160000       zi2: 22.160000       zo1: 0.001119       zo2: 0.001119
zi1: 22.039997       zi2: 24.039997       zo1: 0.000944       zo2: 0.000944
zi1: 23.999996       zi2: 25.999996       zo1: 0.000801       zo2: 0.000801
zi1: 26.039993       zi2: 28.039993       zo1: 0.000685       zo2: 0.000685
zi1: 28.159990       zi2: 30.159990       zo1: 0.000589       zo2: 0.000589
zi1: 30.359989       zi2: 32.359989       zo1: 0.000509       zo2: 0.000509
zi1: 32.639984       zi2: 34.639984       zo1: 0.000442       zo2: 0.000442
zi1: 34.999985       zi2: 36.999985       zo1: 0.000386       zo2: 0.000386
zi1: 37.439980       zi2: 39.439980       zo1: 0.000339       zo2: 0.000339
zi1: 39.959976       zi2: 41.959976       zo1: 0.000298       zo2: 0.000298
zi1: 42.559975       zi2: 44.559975       zo1: 0.000264       zo2: 0.000264
zi1: 45.239971       zi2: 47.239971       zo1: 0.000234       zo2: 0.000234
zi1: 47.999966       zi2: 49.999966       zo1: 0.000208       zo2: 0.000208
zi1: 50.839962       zi2: 52.839962       zo1: 0.000186       zo2: 0.000186
zi1: 53.759960       zi2: 55.759960       zo1: 0.000167       zo2: 0.000167
zi1: 56.759956       zi2: 58.759956       zo1: 0.000150       zo2: 0.000150
zi1: 59.839951       zi2: 61.839951       zo1: 0.000135       zo2: 0.000135
zi1: 62.999947       zi2: 64.999947       zo1: 0.000122       zo2: 0.000122
zi1: 66.239952       zi2: 68.239952       zo1: 0.000111       zo2: 0.000111
zi1: 69.559944       zi2: 71.559944       zo1: 0.000100       zo2: 0.000100
zi1: 72.959938       zi2: 74.959938       zo1: 0.000091       zo2: 0.000091
zi1: 76.439934       zi2: 78.439934       zo1: 0.000083       zo2: 0.000083
zi1: 79.999931       zi2: 81.999931       zo1: 0.000076       zo2: 0.000076
zi1: 83.639923       zi2: 85.639923       zo1: 0.000070       zo2: 0.000070
zi1: 87.359924       zi2: 89.359924       zo1: 0.000064       zo2: 0.000064
zi1: 91.159912       zi2: 93.159912       zo1: 0.000059       zo2: 0.000059
zi1: 95.039909       zi2: 97.039909       zo1: 0.000054       zo2: 0.000054
Press any key to continue . . .
```

منابع:

- اسلاید های درس
- کلاس درس
- فایل های کمکی گذاشته شده به همراه صورت پروژه
- منابع دیگر در اختیار گذاشته شده مانند کتاب و ...