

****

**زیربرنامه:**

hybrid\_greedy\_algorithm\_3D\_2

|  |  |  |
| --- | --- | --- |
| **توسعه دهندگان** | مرتضی نامور |  |
| محمد امین ذوالجناحی |  |
| **تهیه کنندگان مستند** | مرتضی نامور، محمد امین ذوالجناحی | |
| **تاییدکنندگان** | مرتضی نامور | |
| **تاریخ تنظیم سند** | 7/7/1395 | |
| **شناسه سند** | **MC5F031F1** | |
| **زبان برنامه‌نویسی** | **Fortran 90** | |

1. وظایف

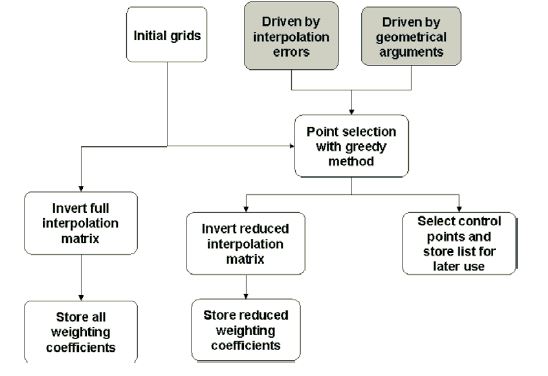
در این زیر برنامه لیستی از نقاط منتخب به همراه تعداد نقاط و ضرایب درونیابی آن ها با توجه به گام اول جابجایی سطوح مرزی توسط الگوریتم Hybrid Greedy 2محاسبه شده و به عنوان خروجی ارائه می شود.

1. توضیحات و تئوری

یک الگوریتم گریدی [1]–[3] به این صورت است که همواره انتخاب های بهینه بر اساس تنها سنجش موضعی از موقعیت دارد. روش Greedy نام دارد چون در هر نقطه این الگریتم فقط بیشترین خطا را برای تصحیح مورد هدف قرار می دهد و عواقب این انتخاب را در نظر نمی گیرد. (ترجمه محض، نامفهوم) در بحث الگوریتم های RBF روش Greedy مورد استفاده قرار می گیرند چون این روش همزمان به حل چالش های محاسباتی تحلیل و به روزرسانی درونیابی کمک می کند. به محض اتمام، این الگوریتم تعدادی از نقاط منتخب تقلیل یافته را همراه با ضرایب مرتبطشان بر می گرداند که باعث تولید مجدد تابع در محدوده ی قابل قبولی از تلرانس می شود. چون درونیابی RBF یک نوع مساله ی جمع کننده نقاط است، اگر همه نقاط انتخاب شده باشند آنگاه خطا در هرجا قاعدتا باید صفر باشد. روش الگوریتم Greedy برای درونیابی RBF می تواند در فرم ها وشکل های مختلف در نظر گرفته شود. تنها تفاوت این روش ها به نوع تولید ضرایب مربوطه، طرز تولید و استفاده از لیستی از نقاط منتخب به نام Active list و نوع پیام خطای تولیدی است که منجر به انتخاب نقاط منتخب می شود. نوع رفتار این روش می تواند به عنوان مثال بوسیله ی تغییر تعداد نقاط اضافه شده به لیست منتخب در هر اجرای حلقه و یا نقاط ابتدایی استفاده شده برای شروع تحلیل در الگوریتم، کنترل شود. در ابتدای امر توصیفی از پیام های خطای نقاط منتخب داده می شود و بعد الگوریتم Hybrid Greedy 2توصیف می­گردد.

* 1. توصیف پیام خطا

اولین سوالی که مطرح است این است که چه نوعی از پیام خطا استفاده شود که بوسیله آن بتوان انتخاب های روش Greedy رو کنترل کرد. این امکان هست که از روش های: آرگومان های هندسی[4]، خطاهای درونیابی[5]، تابع قدرت[4] یا هر نوع دیگری از تابع توصیف شده دیگر برای این هدف استفاده کرد. ‏شکل (1) بیانگر این موضوع است.



1. فلوچارت روش‌هایی برای حرکت شبکه قبل از شروع حلقه تکرار

هنگام استفاده از خطای درونیابی ممکن است روشی را انتخاب کرد که از اطلاعات مستقل باشد؛ که این امر سوال جالبی را که آیا نقاط انتخاب شده باید وابسته به تابع اصلی باشند یا نه، ایجاد می کند. خطاهای  و  و  در جهات *x* و *y* و *z* از تغییر شکل واقعی سطوح بدست می آیند و مقادیر نسبتا قابل قبولی را تولید می کنند. هر کدام از این خطاها در هر نقطه می توند طبق روابط زیر تعریف شود:

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |

که *s* بیانگر جابجایی دقیق نقاط منتخب و زیراندیس *eva(red)* بیانگر محاسبه درونیابی نقاط سطح با استفاده از نقاط منتخب تقلیل یافته در لیست منتخب است. بنابراین فرمول محاسبه خطای واحد در هر نقطه از سطح شبکه توسط رابطه زیر تعریف می گردد:

|  |  |
| --- | --- |
|  |  |

* 1. الگوریتم Hybrid Greedy 2

الگوریتم متعارف RBF از تمامی نقاط سطحی برای محاسبه ضرایب وزنی برای تغییر موقعیت نقاط شبکه ی حجمی استفاده می کند ، این درحالیست که هزینه های محاسباتی بالا برای تغییر موقعیت تمام نقاط شبکه حجمی با استفاده از نقاط سطحی بسیار بالاست. برای کاهش هزینه های محاسباتی تعداد بسیار کمتری از نقاط سطحی باید انتخاب شوند و به عنوان نقاط سطحی پایه در نظر گرفته شوند بطوری که از دقت حل به طور قابل توجه کم نشود. برای نیل به این هدف الگوریتم Hybrid Greedy 2 به صورت زیر تعریف می­شود:

* 1. الگوریتم اول گریدی (First Greedy 2)

در صورتی که اندیس هر نقطه ی مرزی باشد داریم:

|  |  |
| --- | --- |
|  |  |
|  |  |

با پیدا کردن ثابت خواسته شده، نقطه انتخابی تصحیح می شود:

|  |  |
| --- | --- |
|  |  |

بردار ضرایب نیز تصحیح می گردد:

|  |  |
| --- | --- |
|  |  |

مقدار درونیابی در تمامی نقاط تصحیح می شود، که  همان ارزیابی درونیابی است:

|  |  |
| --- | --- |
|  |  |

خطاهای جدید در این قسمت محاسبه می شوند، که  همان مقدار دقیق تابع است:

|  |  |
| --- | --- |
|  |  |

در مرحله ی بعد مقدار خطای ماکسیمم تعیین می شود:

|  |  |
| --- | --- |
|  |  |
|  |  |

بازگشت به رابطه ‏(7) و یا اگر مراحل به اتمام رسیدند ، نقاط منتخب در لیست مربوطه ذخیره می شوند و در گام های بعد دقیقا جابجایی نقاط داخلی و نقاط غیر منتخب شبکه نسبت به این نقاط منتخب انجام می شود.

لازم است ذکر شود که تصحیح می تواند برای یک نقطه و یا برای چند نقطه مرزی غیر منتخب در هر سیکل انجام شود[6] و[1] و[7]. اگر چه اضافه کردن نقاط جایی که خطا بزرگ است ایده ی ساده ای به نظر می­رسد اما این پیچیده تر از آن است ک بتوان تصمیم گرفت چطور نقاط غیر ضروری را از حذف کرد. این امر به این دلیل است که اگر چه لیستی از نقاط موجود است که خطا در آن ها بزرگ است ، اما نقاطی که استفاده شده اند یا خطای فر دارند یا خطایشان بسیار کوچک است و همین امر باعث سختی و هزینه بر بودن تشخیص بین این نقاط می شود. الگوریتم First Greedy 2 تمایل به نشان دادن درجه ای از رفتار خود محدود کنی درباره اینکه چه تعداد نقطه استفاده می کند، دارد و اغلب یک نقطه ی قبلی تعیین شده را تصحیح می کند به جای اینکه نقطه جدیدی معرفی کند. این امر مزیتی ندارد چون به عدم بهینه بودن کافی روش در حل دستگاه معادلات خطی اشاره دارد یعنی اگر چه تعداد نقاط رفته رفته به آرامی افزایش می یابند، پیام خطا نیز رفته رفته و به آرامی کم می شود. الگوریتم First Greedy2 خیلی سریع است؛ احتمالا سریع ترین الگوریتم های روش RBF که در زمانی متناسب با تعداد نقاط  به انجام می رسد. متاسفانه مانند روش خالص تابع پایه شعاعی برای حل مسائل درونیابی، ضرر هایی متحمل این روش نیز هست. در ابتدا روند همگرایی بسیار سریع است اما سریعا از سرعت آن هنگامی که نقاط اضافه می شوند کاسته می شود. ارتباط بین سرعت و تعداد تکرار های حل به صورت  که *k* همان شمارنده تکرار حلقه است، بیان می­شود. الگوریتم دوم Greedy به جای حل و تصحیح یک تک نقطه در زمان، یک سیستم کامل را قبل از اینکه تصمیم گرفته شود چه نقطه ای باید به لیست اضافه شود، حل می کند.

* 1. الگوریتم دوم گریدی (Second Greedy)

هدایت اولین الگوریتم Greedy به یک تحلیل کامل مساله درونیابی در هر گام زمانی، به این معنی است که انتخاب نقاط بوسیله ی تنها یک روند تصحیح کننده در هر نقطه محلی تاثیر نمی پذیرد.(همانطور که در الگوریتم اول توضیح داده شد). همچنین این امکان موجود است که این روش را با اضافه کردن بیشتر از یک نقطه به لیست منتخب در یک زمان انجام داد. در این مورد بسیار ضروریست که یک مجموعه و قوانینی تشکیل داد که بر مبنای مرتب کردن نقاط شکل گرفته باشند تا از آن طریق بتوان نقاط با بیشترین خطا را انتخاب کرد. روند الگوریتم به اینصورت است:

1-در وحله اول مجموعه نقاط مرزی غیر منتخب که دارای بیشترین خطا هستند انتخاب می شوند:

|  |  |
| --- | --- |
|  |  |

2-با توجه به نقاط انتخابی و نقاط داخل لیست منتخب از الگوریتم اول و جابجایی دقیق مجموعه این نقاط مرزی، ضرایب درونیابی آنها تعیین می شود.

3-درونیابی تمام نقاط مرزی غیر منتخب نسبت به مجموعه نقاط منتخب ارزیابی می شود:

|  |  |
| --- | --- |
|  |  |

4-خطاها پیدا می شوند:

|  |  |
| --- | --- |
|  |  |

5-شامل نقاط جدید با بزرگترین مقدار خطا و برگشت به مرحله 2(در صورت بیش از یک بار فراخوانی).

نوع دیگری از الگوریتم Greedy ترکیبی از این دو روش اول و دوم است تا یک روش Hybrid ایجاد شود به اینصورت که روند سریع اضافه کردن نقطه در الگوریتم اول می تواند با بازده نقطه برتر الگوریتم دو ترکیب شود و الگوریتم سومی به نام Hybrid Greedy پدید آورد. برای این روش سیکل ها در حال تغییر اند به طوریکه الگوریتم اول می تواند بیشتر اوقات به خاطر سرعتش اجرا شود، اما الگوریتم دوم در هر *m* تکرار حلقه فراخوانی می شود.

* 1. اجرا در برنامه

برای وضوح بیشتر نحوه اجرای روش Hybrid Greedy 2 در یک کد سعی می‌گردد تا آنچه که تاکنون به عنوان مفاهیم ریاضی روش بیان شده است، به صورت جزئی تر بیان گردد تا متن وضوح بیشتری داشته باشد.

در مسائل دینامیک سیالات محاسباتی، دامنه محاسباتی به دو بخش نواحی داخلی و مرزی تقسیم می‌گردند. روش‌های اصلاح شبکه برپایه‌ی تغییرات مختصات نقاط مرزی قرار گرفته اند، به عبارت دیگر در همه‌ی این روش‌ها باید در ابتدا جابجایی نقاط منتخب مرز به عنوان یک معلوم تعیین گردد و در گام بعد جابجایی نقاط دیگر مرزی محاسبه می‌شوند. در الگوریتم Hybrid Greedy 2 ضرایب نقاط منتخب و مقادیر درونیابی در نقاط مرزی منتخب هر دو تصحیح می شوند و روند به این صورت است که در هر 10 تکرار حلقه، الگوریتم دوم یکبار فراخانی می شود.

بردار جابجایی نقاط مرزی  شامل مقادیر معلوم جابجایی برای نقاط گسسته می‌باشد. تابع میانیابی ، قابلیت محاسبه‌ی جابجایی نقاط (اعم از داخلی و مرزی) را دارا می‌باشد. بنابراین همان طور که بیان شد: . در اینجا  همان نقاط انتخابی الگوریتم ست.

اگر تمام نقاط مرزی منتخب را با اندیس *s* نامگذاری کنیم، بر اساس رابطه زیر وجود دارد:

|  |  |
| --- | --- |
|  |  |

که در این رابطه *NSBP* تعداد نقاط انتخاب شده و  ضرایب آن نقاط انتخابی و  همان مکان نقاط انتخابی الگوریتم است. همانطور که می دانیم جابجایی دقیق نقاط انتخابی مرز ها موجود است پس می توانیم به جای عبارت  برای هر نقطه ی منتخب مقدار  را قرار دهیم.

بر اساس رابطه ‏(16)، اگر تعداد نقاط مرزی منتخب برابر باشد با  پس به تعداد  معادله وجود دارد:

|  |  |
| --- | --- |
|  |  |

باید توجه شود که  ، به اشاره دارد.

در رابطه بالا مجهولات عبارتند از بردار ضرایب  که عبارت است از  که  عضو دارد.

حال دستگاه معادلاتی به شکل رابطه زیر حاصل می‌گردد:

|  |  |
| --- | --- |
|  |  |

که:

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |

ضرایب بدست آمده *Cox* و *Coy* نیز توسط رابطه ‏(8) تصحیح می­شوند و به صورت *Coxx* و *Coyy* از برنامه خارج می شوند. بدین ترتیب ضرایب نقاط منتخب لیست نقاط مرزی با استفاده از جابجایی دقیق خودشان طبق رابطه ‏(5) بدست می آیند. گام بعد محاسبه ی جابجایی تمامی نقاط غیر منتخب مرزی با استفاده از نقاط منتخب مرزی است بنابراین داریم:

|  |  |
| --- | --- |
|  |  |
|  |  |

که در روابط بالا اندیس*ns* مربوط به نقاط مرزی غیر منتخب می باشد. متغیر *k* بیانگر شمارنده تکرار حلقه است. در مرحله بعد مقادیر درونیابی  و  توسط رابطه ‏(9) تصحیح می شوند یعنی :

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

در گام بعد خطا ها محاسبه می شوند بنابراین داریم:

|  |  |
| --- | --- |
|  |  |
|  |  |

و نهایتا خطای واحد طبق رابطه زیر تعیین می گردد:

|  |  |
| --- | --- |
|  |  |

در

|  |  |
| --- | --- |
|  | Add point to active list |

که در این رابطه برابر خطا و  بیشترین مقدار خطا در تکرار قبلی و  پارامتری است که محدوده انتخاب نقاط را تعیین می کند. سپس مجموعه نقاط انتخابی توسط الگوریتم دوم به مجموعه لیست منتخب نقاط مرزی اضافه شده و با استفاده از معادلات ‏(17) تا ‏(21) ضرایب *Cox* و *Coy* و *Coz* تعیین می­شود. آنگاه مقدار تابع درونیابی برای تمام نقاط مرزی غیر منتخب به ازای نقاط انتخاب شده توسط هر دو الگوریتم توسط رابطه‌های ‏(22) و ‏(23) محاسبه شده و خطا ها طبق روابط ‏(29) تا ‏(31) محاسبه می گردند. نهایتا بیشینه خطا برابر است با :

|  |  |
| --- | --- |
|  |  |

در صورت ارضا نشدن شرط حلقه کلی، از این خطاها برای تعیین ضرایب تصحیح  و  و در الگوریتم اول استفاده می شود.

1. بخش‌های زیربرنامه

در این قسمت، توضیح تمامی بخش‌های زیربرنامه، مطابق شماره‌گذاری انجام شده در متن برنامه کامپیوتری ارائه شده است.

بطور خلاصه می توان گفت که در روش Hybrid Greedy2 هدف به دست آوردن ضرایب تابع میانیابی  می‌باشد که باید ضرایب مجهول آن یعنی  مشخص شوند. این ضرایب با حل شدن دستگاه معادلات  بدست می آیند که  به ماتریس ضرایب اختصاص دارد و نهایتا ضرایب خروجی از برنامه و مقادیر درونیابی برای محاسبه بیشینه خطا تصحیح می شوند.

1. انتخاب یک نقطه تصادفی و مقدار دهی اولیه

در ابتدا یک نقطه مرزی به طور تصادفی انتخاب می گردد تا الگوریتم شروع به کار کند؛ خطا نیز مقدار دهی اولیه می شود. مقادیر و  و  با استفاده از مقدار دقیق جابجایی نقطه انتخابی مقدار دهی اولیه می­شوند.

1. تعریف محدوده ی تکرار حلقه کلی

حلقه ابتدایی برای متوقف کردن روند کار الگوریتم First Greedy2 با توجه به محدوده مطلوب خطا تعیین می شود. که در این قسمت این محدوده برای خطاهای بزرگتر از  تنظیم شده است.

1. تعیین مقدار  و  و و 

در این قسمت با فراخوانی تابع پایه ای شعاعی مقدار و و  وطبق رابطه ‏(24) و ‏(25) محاسبه می­شوند و به تعداد نقاط منتخب اضافه می شود.

1. تعیین شرط عدم تعرض

برای اینکه تعداد مقادیر نقاط مرزی منتخب از تعداد کل نقاط مرزی تعدی نکند، یک شرطی با دستور Goto نوشته می شود و آن نقطه منتخب در لیست قرار می گیرد.

1. تعیین ماتریس مقادیر *b*

ماتریس جابجایی های دقیق طبق رابطه ‏(19) باتوجه به نقطه منتخب در راستای x و y وz تعیین می­شود.

1. محاسبه ماتریس *B*

در این قسمت نیز ماتریس *B* با استفاده از زیر برنامه ی مربوطه به صورت رابطه ‏(20) محاسبه می شود.

1. محاسبه ماتریس ضرایب درونیابی

با استفاده از حلگر LU و ماتریس های بدست آمده از دو مرحله قبل، ماتریس ضرایب تابع درونیابی نقاط منتخب در این قسمت محاسبه می شود.

1. تصحیح ضرایب نقاط منتخب تقلیل یافته در لیست منتخب

در یک حلقه تکرار ضرایب تمامی نقاط لیست منتخب بدست آمده از مرحله قبل توسط **رابطه ‏(26) تصحیح می شوند.**

1. مقدار دهی اولیه خطای میانگین و بیشینه خطای کل

مقدار دهی اولیه خطای میانگین و خطای بیشینه کل در این قسمت صورت می گیرد.

1. حلقه محاسبه کننده جابجایی تمام نقاط غیر منتخب

در این بخش بر اساس معادلات ‏(22) و ‏(23) جابجایی تمام نقاط مرزی غیر منتخب مشخص می‌گردد.

1. ذخیره مختصات نقطه مورد بررسی در پارامترهای محلی

در این قسمت مختصات نقطه مورد بررسی در پارامترهای محلی ذخیره می گردد تا در مراحل بعدی از آنها استفاده گردد.

1. مقدار دهی اولیه

همانگونه که از معادلات ‏(22) و ‏(23) مشخص است، بخشی از این روابط بصورت مجموع عبارت هایی می باشد که در ابتدای حلقه آنها را برابر صفر قرار می دهیم.

1. محاسبه‌ی 

در این حلقه بخش  از تابع میانیابی  برای هر نقطه مرزی غیرمنتخب محاسبه می‌شود. در واقع این حلقه کار عملگر  را انجام می‌دهد.

1. انصراف از محاسبه نقاط مرزی منتخب

از آنجا که مقدار جابجایی نقاط مرزی منتخب معلوم است، نیازی به محاسبه آن نمی باشد که برای این منظور از دستور Goto استفاده می شود.

1. محاسبه فاصله

با توجه به اینکه تابع  خود تابعی از فاصله تا نقاط مرزی می باشد، در اینجا این فاصله نقاط مرزی منتخب از نقاط مرزی غیر منتخب محاسبه می گردد.

1. محاسبه تابع 

باید توجه داشت که تابع  توسط زیر برنامه RBF\_Function\_3D فراخوانی می‌گردد.

1. محاسبه بخشی از تابع میانیابی

در این بخش مقادیر زیر محاسبه و به مقادیر قبلی اضافه می گردد:

|  |  |
| --- | --- |
|  |  |

1. محاسبه 

مقدار این رابطه در این بخش از برنامه محاسبه می شود.

1. تصحیح مقادیر محاسبه شده ی درون یابی

در این بخش مقادیر محاسبه شده درونیابی توسط روابط ‏(27) و ‏(28) تصحیح می شوند.

1. محاسبه خطاها

خطای محاسبات در این بخش توسط روابط ‏(29)، ‏(30) و ‏(31) تعیین می گردند.

1. تعیین ماکسیمم مقدار خطا و ذخیره آن

در این بخش ماکسیمم مقدار خطای بدست آمده در هر حلقه برای مجموعه نقاط مرزی غیر منتخب تعیین می گردد و اندیس آن نیز مشخص می گردد و در پارامتر محلی ذخیره می شود.

1. محاسبه خطای میانگین

ابتدا تعداد نقاط مرزی غیر منتخب در یک پارامتر محلی ذخیره می شود و خطای مینگین طبق رابطه مربوطه محاسبه می گردد.

1. شرط اعمال الگوریتم دوم Greedy

در این بخش اگر شمارنده تکرار حلقه مضربی از *m* باشد، الگوریتم دوم فعال می شود.

1. پیدا کردن نقاط مرزی با محدوده خطای بالا

با تعریف یک حلقه به تعداد نقاط مرزی غیر منتخب در ابتدا نقاط مرزی منتخب از بین کل نقاط مرزی فیلتر می شوند و غیر منتخب ها برای اینکه آیا در محدوده ی خطای بالا قرار بگیرند یا نه، کنترل می شوند.

1. اعمال شرط الگوریتم دوم Greedy

با اعمال یک دستور شرطی طبق رابطه ‏(32) نقاط درون محدوده ی خطای تعیین شده انتخاب می­شوند.اندیس نقاط برداشته می شود و به تعداد نقاط منتخب اضافه می گردد.

1. شرط عدم تعرض نقاط منتخب از مجموعه نقاط مرزی

در این قسمت شرط عدم تعرض نقاط منتخب از مجموعه نقاط مرزی اعمال می شود و شمارنده نقاط منتخب توسط الگوریتم دوم چاپ می شود و نهایتا به لیست نقاط منتخب اضافه می شود.

1. تشکیل ماتریس *b* برای محاسبه ضرایب تابع درونیابی

با توجه به نقاط انتخابی توسط الگوریتم دوم و اضافه کردن آن ها در لیست نقاط منتخب توسط الگوریتم اول، ماتریس b طبق رابطه ‏(19) در دو راستای متفاوت x, y,z تشکیل می شود.

1. محاسبه ماتریس *B*

در این قسمت نیز ماتریس *B* با استفاده از زیر برنامه­ی مربطه به صورت رابطه ‏(20) محاسبه می شود.

1. محاسبه ماتریس ضرایب درونیابی

با استفاده از حلگر LU و ماتریس های بدست آمده از دو مرحله قبل، ماتریس ضرایب تابع درونیابی نقاط منتخب در این قسمت محاسبه می شود.

1. مقدار دهی اولیه به خطای بیشینه و شروع درونیابی

در این قسمت به تابع بیشینه کل مقدار دهی اولیه می شود و سپس درونیابی نقاط مررزی غیر منتخب بوسیله نقاط مرزی منتخب حاصل از هر دو الگوریتم شروع می شود.

1. ذخیره مختصات نقطه مورد بررسی در پارامترهای محلی

در این قسمت مختصات نقطه مورد بررسی در پارامترهای محلی ذخیره می گردد تا در مراحل بعدی از آنها استفاده گردد.

1. مقدار دهی اولیه

همانگونه که از روابط ‏(22) و ‏(23) مشخص است، بخشی از این روابط بصورت مجموع عبارت هایی می باشد که در ابتدای حلقه آنها را برابر صفر قرار می دهیم.

1. محاسبه‌ی 

در این حلقه بخش  از تابع میانیابی  برای هر نقطه مرزی غیرمنتخب محاسبه می‌شود. در واقع این حلقه کار عملگر  را انجام می‌دهد.

1. انصراف از محاسبه نقاط مرزی منتخب

از آنجا که مقدار جابجایی نقاط مرزی منتخب معلوم است، نیازی به محاسبه آن نمی باشد که برای این منظور از دستور Goto استفاده می شود.

1. محاسبه فاصله

با توجه به اینکه تابع  خود تابعی از فاصله تا نقاط مرزی می باشد، در اینجا این فاصله نقاط مرزی منتخب از نقاط مرزی غیر منتخب محاسبه می گردد.

1. محاسبه تابع 

باید توجه داشت که تابع  توسط زیر برنامه RBF\_Function\_2D فراخوانی می‌گردد.

1. محاسبه بخشی از تابع میانیابی

در این بخش مقادیر زیر محاسبه و به مقادیر قبلی اضافه می گردد:

|  |  |
| --- | --- |
|  |  |

1. ذخیره مقادیر درونیابی در بردار محلی

بدون توضیح.

1. محاسبه خطاها

خطای محاسبات در این بخش توسط روابط ‏(29)، ‏(30) و ‏(31) تعیین می گردند.

1. تعیین ماکسیمم مقدار خطا و ذخیره آن

در این بخش ماکسیمم مقدار خطای بدست آمده در هر حلقه برای مجموعه نقاط مرزی غیر منتخب تعیین می گردد و اندیس آن نیز مشخص می گردد و در پارامتر محلی ذخیره می شود.

1. چاپ نتایج خطا

مقادیر خطاهای محاسبه شده ائم از ماکسیمم خطا و خطای میانگین نسبت به تعداد نقاط انتخابی مرز و تکرار حلقه چاپ می شوند.

1. ذخیره خطای بیشینه در پارامتر محلی

ماکسیمم خطای بدست آمده در حلقه قبل در پارامتر محلی ذخیره می شود.

1. تعیین مقادیر  و و

در صورتی که مقدار ماکسیمم خطای بدست آمده شرط حلقه کل را ارضا نکند مقادیر  و  و  در این بخش برای استفاده در تکرار بعدی محاسبه می شوند که این عبارات همان صورت روابط ‏(21) و ‏(22) هستند.

1. مراجع

[1] R. Schaback and H. Wendland, “Adaptive greedy techniques for approximate solution of large RBF systems,” *Numer. Algorithms*, 2000.

[2] Y. Ohtake, A. Belyaev, and H. Seidel, “Multi-scale and adaptive cs-rbfs for shape reconstruction from clouds of points,” *Adv. Multiresolution*, 2005.

[3] R. Schaback and H. Wendland, “Numerical techniques based on radial basis functions,” 2000.

[4] S. De Marchi, “On optimal locations for radial basis function interpolation: computational aspects,” *Rend. Del Semin. Mat.*, 2003.

[5] J. Carr, R. Beatson, J. Cherrie, and T. Mitchell, “Reconstruction and representation of 3D objects with radial basis functions,” *Proc. 28th*, 2001.