التحليل العددي

Numerical Analysis

عمر محمد التومي
أحمد حمر الشوشة

منتديات التقنية
WWW.TKNE.NET

المحتويات

الموضوع ******************** الصفحة

المحتويات
كلمة المنتدى
عزيزي القارئ
الباب الأول: مقدمة في التحليل العددي
1-1 استهلال 6

2-1 0.4 MICROSOF DEVELOPER STUDIO 7

الباب الثاني: طرق حل العادلات ذات المجهول الواحد

1-2 طرقة المقاطع 11

2-2 طريقة نيوتن 16

3-2 طريقة نيوتن رافسن ونيوتن رافسن المعدلة 19

4-2 طريقة الموقع الخاطئ 42

الباب الثالث : الحلول العددية للمعادلات التفاضلية العادية
1-3 الحلول العددية للمعادلات التفاضلية العادية 44
2-3 طريقة اويلر 45
3-3 طريقة اويلر الاكثر امتدادا 45
4-3 طريقة اويلر المعدلة 45
5-3 طريقة رنج كوتا 46
6-3 طريقة الرمي 54

الباب الرابع : الحل العددي للمعدلات لنظام المعدلات الخطية

1-4 طريقة جاكوبي لحل مسائل القيم الذاتية 57

2-4 طريقة لا جرانج للتوليد 67

3-4 حذف جاوس: طريقة التعويض الخلفي 76

الباب الخامس الملاءمة والانكفاء بواسطة البرمجة

1-5 الملاءمة و الانكفاء 86

الخاتمة 98

كلمة مدير الموقع

بسم الله الرحمن الرحيم

الحمدلله والصلاة والسلام على أشرف الخلق والمرسلين سيدنا محمد صلى الله عليه وسلم ..
إن المتابع الجيد للساحة العربية على الشبكة العنكبوتية ليجد أنها تفتقر الى الكثير من المواضيع التي تهم المتخصصين كالمهندسين والفنيين والتقنيين عموماً، وقد نجد في بعض المواقع بعض الاجتهادات من توفير المراجع ولكنها في الغالب تكون مصادرَ أجنبية وبلغات غير اللغة العربية.

من هذا الباب .. عكف فريق موقع و منتديات التقنية (أكبر تجمع للمهندسين العرب) www.tkne.net على تبني مشروع بناء مكتبة عربية متخصصة في المواضيع الهندسية والتقنية ، بهدف نشر هذه الكتب وتوزيعها الكترونياً عبر الموقع لتساهم ولو بجزء بسيط في اثراء الساحة العربية بالمواضيع العربية الهندسية.

وهذا الكتاب الالكتروني الذي بين يديك هو نتاج هذا المشروع الجميل ، الذي نسأل الله أن يمدنا بالعون والتوفيق على اكماله، والشكر لجميع الاخوة ممن ساهم في اخراج هذا الكتاب أو ساعدنا بالرأي والمشورة والاقتراح وصلى الله وسلم على سيدنا محمد وعلى آله وصحبه وسلم .

بقلم المهندس/ فهد الرفاعي
أستاذ في كلية جازان السعودية
والمشرف العام لموقع التقنية
refaeefa@hotmail.com

عزيزي القارئ ..
بين يديك كتاب يُعنى بالتحليل العددي ,هدفه عرض رياضيات الحاسب الآلي من زوايا شتى تتيح لنا التعرف على آفاق مختلفة في هذا المجال .
ومن جهة أخرى يسعى الكتاب لترسيخ مفهوم العديد من المعادلات والبرامج والقوانين ذات الصلة بالتحليل العددي مما يجعل من الكتاب مفيداً للطلاب وللمبرمجين على حد سواء .
سبق ونُشرت مادة هذا الكتاب ضمن منتدى التقنية , وها هو الموقع يقدمها لكم في إطار الكتروني ضمن مشروع المكتبة المجانية وبأسلوب سلس يجعل مهمة القارئ يسيرة في القراءة .
نتمنى لكم وقتاً طيباً وفائدةً جمة .

الباب الأول

مقدمة في التحليل العددي

1-1 استهلال introduction

التسمية الدقيقة لهذا التخصص هو التحليل العددي(numerical analyses) يستخدم التحليل العددي في حل المعادلات الرياضية التي يصعب حلها أو يستلزم وقت طويلا في الحل، فإذا كانت لدينا معادلة رياضية من الدرجة الرابعة أو الخامسة فان من الممكن حلها بالطرق التقليدية، و حتى هذه الطرق تاخد وقتا، إما إذا كانت المعادلة من الدرجة السادسة و أكثر يكون من الصعب دائما التعامل مع المعادلة تقليديا، و مع ظهور الحاسب الآلي (الكومبيوتر) اتضحت أهميته البالغة في حل هكذا معادلات، و ذلك لتوفيره الوقت و الجهد. و بالأخص في المعادلات التي تحتاج إلى تكرار كبير من اجل الوصول إلى النتيجة أو الحل.

عندما يحصل فني أو مهندس على وظيفة معينة، مثلا في الصيانة أو الإنتاج أو الجودة أو تدبير المواد الأولية، يتوقع أنه سيفتح جهاز الحاسوب ليجد أمامه أحد برامج SAP أو Baan أو PeopleSoft أو Oracle أو JDEdwards و كل ما عليه هو الضغط على بضعة أزرار هنا و هناك ليعرف حالة المخزون و الطلبيات و موعد تسليمها و نتائج اختبارات الجودة و كمية المواد الأولية التي يجب شراؤها و عدد الـ lots في خطوط الإنتاج و أسماء العمال الذين عملوا في تجميع منتج معين و عدد ساعات غياب عامل ما و الاقتراحات التي قدمها العمال و عدد توقفات آلة ما ...لكن هذا غير موجود على أرض الواقع، واقعنا العربي...

و حتى الشركات الغنية كشركات البترول و الصلب، التي استثمرت في هذه الحلول و ركبتها، لا تستفيد منها كاملا و أغلب الموظفين يعتبرونها عبئا إضافيا على واجباتهم اليومية و قد يستعملونها فقط عند قدوم زوار خارجيين. أما في الشركات المتوسطة و الصغيرة، فمن الصعب على الفني أو المهندس إقناع المدراء بالفائدة الملموسة لمثل هذه الأدوات خصوصا أن ثمن تملكها السنوي Total Cost of Ownership مرتفع مقارنة بميزانية الشركة السنوية، لذلك

فالحل الوحيد الذي يجده المهندس لتدبير أموره اليومية هو ابتكار حلول برمجية على مقاس الشركة .

و لن يتأتى لـه ذلك إلا بمعرفة مسبقة بطرق التحليل العددي و استعمال الحاسوب في حل معادلات رياضية. و هذا يدرس في الجامعات و مراكز التكوين (و منتدانا الحبيب) و ليس في مكان العمل. لأن وقت العمل ضيق و محسوب.كما أنـه لا مجال لاستعمال برامج مقرصنة تحت طائلة تعريض الشركة لغرامات مالية في حال قيام مايكروسوفت و أخواتها بالتحقق من برمجيات الشركة (إثر مكالمة هاتفية من عامل لم يستفد من العلاوة السنوية).

قد تكون الصورة التي رسمتها قاتمة لكنها واقعية و تبين أن البرمجيات التي تستعملها في حاسوب المنزل لا تستطيع تثبيتها كلها في حاسوب العمل دون رخص.

في النهاية، أقول إني لا أقصد أن يكون المهندس مبرمجا محترفا زيادة على التخصص الذي درسه، لكن وضع برنامج صغير لتتبع مؤشر ما او ورقة excel ببعض المعادلات الرياضية أسرع و أنجع من تعلم برامج من العيار الكبير في حالات كثيرة.

في حقيقة الأمر يوجد الكثير من البرامج يمكن استخدامها في هذا المجال مثل برنامج MATHLAB , MATHCAD و حتى برنامج EXCEL المرافق لحزمة OFFECE يمكن أجراءه في حل الكثير من المعدلات و إجراء العديد من الصغير الرياضية المفيدة ، لكن في هذه السلسلة سنقوم باستخدام إحدى لغات البرمجة في التعامل مع المعادلات الرياضية ، سنتعامل مع لغة FORTRAN لما تمتاز به هذه اللغة من استقرار عال و دقة في المعادلات و تحديد دقيق لنوع المتغيرات مع سعة كبيرة في نوع المتغيرات، من الممكن أنها ليست على درجة كبيرة من الانتشار و الاستخدام ، غير أن المتخصصين في مجال التحليل العددي و الرياضيات بصفة عامة يعترفون لها بالفضل، سنستخدم الإصدار الرابع منها و هي

1- 2 MICROSOF DEVELOPER STUDIO 0.4

و هو إصدار يعمل تحت بيئة الويندوز(WINDOWS) و لست في معرض الدعاية لهذه اللغة و لا أحب المقارنة بين اللغات المختلفة غير أن FORTRAN LANGUAGE هي لغة المهندسين بكل كفاءة.

غير أن هذا الكتاب لن يهمل بعض اللغات ذات الانتشار الواسع مثل لغة C و قد اعتمدت بعض الطرق المدرجة في هذا الكتاب بلغة C من اجل خلق تنوع يفيد

القارئ و يثري أفكاره، مع ملاحظة وجود الخوارزمية و مخطط سير العمليات لمعظم البرامج المدرجة، و تم تحليل البيانات و التعليق عليها في البعض الأخر، و توجد رسومات توضيحية باستخدام math lab لمحاولة تغطية اكبر قدر ممكن من الجوانب الهامة في هذا الموضوع.

قبل التعامل مع المعدلات الرياضية لا بد أن نقدم أساسيات لغة FORTRAN من التواب و المتغيرات.

أساسيات لغة فورترانFORTRAN

1 الرموز CHARACTERS

تستخدم مجموعة من الرموز الأساسية في لغة FORTRAN و تتكون من الأتي :

1.1 الأرقام NUMBERS CHARACTERS

و هي 0 1 2 3 4 5 6 7 8 9

1.2 الحروف ALPHABATIC CHARACTERS

تشمل الحروف المستخدمة في لغة الإنجليزية و هي :

A B C ....X Y Z

1.3 رموز خاصة SPECIEL CHARACTERS

و هي :

المساواة الفارزة

النقطة العشرية

القوس الأيمن)

القوس الأيسر (

الفاصلة العليا (القسمة) ÷ /

النجمة (الضرب) *

الزائد +

الناقص –

2.1 أنواع البيانات في لغة FORTRAN

تمثل البيانات لفي لغة الفورتران FORTRAN بإحدى الأساليب الآتية :

2.2 صحيحة INTEGAR

و تشمل جميع الأعداد

حقيقية RAEL الأعداد التي تحتوي على العلامة العشرية

2.3 مزدوجة الدقة DOUBEL PRECESION

أعداد صحيحة آو حقيقة تخزن بدقة كبيرة

2.4 مركبة COMPLEX

أعداد تحتوي على كل من الجزء الحقيقي و التخيلي RAEL AND IMAGE

2.5 منطقية LOGICAL

القيم المنطقية أو الصادقة

2.6 الحروف الرقمية ALPHANUMERIC

المعلومات اللفظية LITERL INFORNATION الأنواع الربعة الأولى تستخدم لتمثيل القيم العددية خلال العمليات الحسابية ، و الأسلوب المنطقي يمثل القيم الصادقة او الكاذبة (TRUE OR FULSE)

الباب الثاني

طرق حل العادلات ذات المجهول الواحد

SOLUTION OF EQUATION IN ONE VARIABLE

1-2 طريقة المقاطع

BISECTION METHOD

لنفرض أن لدينا المعالة الرياضية التالية :

$$X^2 + 4 = 0$$

مع ملاحظة أن الرمز(^) يعني أس أي مربع القيمة (X) و X^5 يعني الأس الخامس للمتغير X.

سيكون الحل المثالي لهذه المعادلة هو كالأتي:

$$X^2=4$$

$$X=\pm\sqrt{4}$$

$$X=-2$$

X=-2وهو حل دقيق تماما بمجرد إتباع الخطوات السابقة, نحصل على الحل.

و إذا كانت لدينا المعادلة الرياضية التالية :

$$X^3+4X^2-10=0$$

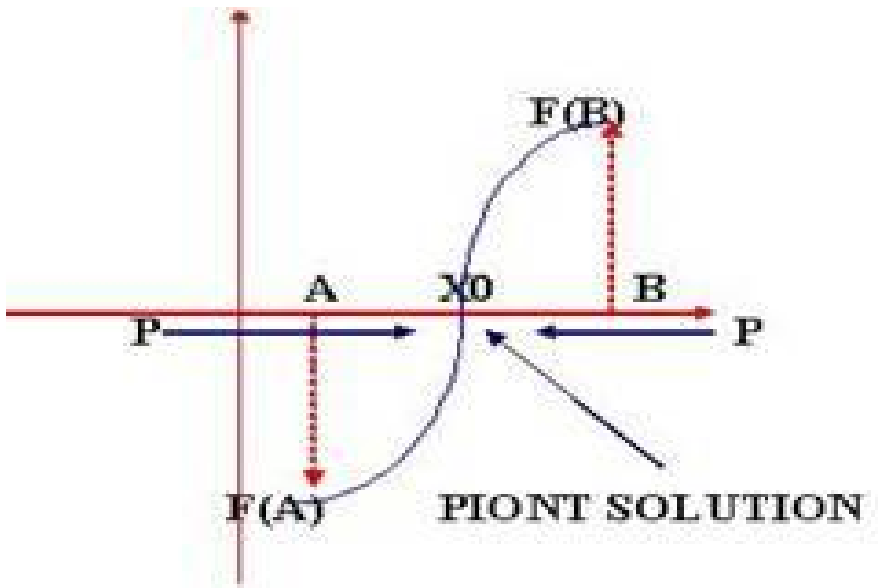
فحلها يرهق قليلا و لا يعطي قيم دقيقة ، و لهذا نلجأ إلى طريقة المقاطع BISECTION METHOD التي تتلخص فيما يأتي :

لنفرض أن لدينا دالة F(X) معرفة و مستمرة خلال الفترة { A , B } مع الدالتين F(A) , F(B) فهنالك قيمة X0 تنتمي إلى الفترة

{A,B} كالدالة F(X0)=0.0 ، و التي تكون حلا للمعادلة F(X)

بما أن القيمتين A B تنتمي إلى نفس الفترة فهذا يعني ان الدالتين

F(A) , F(B) تنتمي إلى نفس النطاق و يكون الرسم البياني للدالة كالآتي:



(شكل 2-1)

و حل المعادلة يكون هو النقطة X0 (PIONT SOLUTION) و بما أن أساس الحل تقريبي فتكون الطريقة المثالية لخطوات الحل هي اخذ مجموعة من النقاط التقريبية المقربة للقيمة الصفرية كالقيمة 0.00001

و اختبارها فهي لا تساوي صفر لكنها دقيقة جدا و مقربة جدا من الصفر حيث يصعب الحصول على الحل الصفري في التحليل العددي مباشرة و هذه القيمة سنفرضها ليحملها المتغير P و يقترب كل مرة بقيمة دقيقة من الحل الصفري و تعتمد طريقة المقاطع (BISECTION METHOD) على نفس الفكرة في التعامل مع المسائل الرياضية و خوارزمية البرنامج هي كالآتي :

START

RAED (A , B)

DO P=(A +B) / 2

Y1= A^3 + 4*A^2 – 10

Y2= B^3 + 4*B^2 -10

Y0=P^3 + 4*P^2 – 10

IF (Y0 < 0.00001) GOTO STEP 100

IF (Y0*Y1 > 0.0) GOTO STEP 10

A=A

B=B

GOTO STEP 20

DO A=-P

B=B

GOTO STEP 20

WRITE Y0 , P

STOP

شرح الخوارزمية :

نبدأ بقراءة القيمتين A B ثم نقوم بتطبيق القيمة المتوسطة للقيمتان من اجل تسريع الوصول للقيمة الصفرية و ذلك بأخذ المتوسط لهما و هي القيمة (P) ثم نقوم بتطبيق الدوال السابقة من اجل اختبارها و يتم ذلك باختبارين

الاختبار الأول يسأل هل قيمة Y0< 0.00001 بمعنى أن الحل قد تحقق اذا كانت الإجابة بنعم و الا فيتم الانتقال إلى الاختبار الثاني الذي يسال هل قيمة مضروب الدالتين Y0,Y1 اكبر من الصفر إذا كانت الإجابة بنعم فتأتي خطوة التقريب الأولى التي تضع قيمة P مكان قيمة B و تثبت قيمة A فيكون الاقتراب من جانب الدالة F(B) نحو القيمة الصفرية التقريبية و الا فيتم الانتقال الى الخطوة التالية التي تقوم بالتعويض عن قيمة A = P و تطبق ذلك على الدوال و تختبر القيم و هكذا يستمر الحل حتى يتحقق الشرط و تقترب الدالة من القيمة الصفرية و يتم طباعة القيمتين P , Y0

البرنامج بلغة FORTRAN

بعد تنصيب البرنامج تقوم بفتحه و تتبع الخطوات التالية:

1. **من القائمة المنسدلة FILE تختار NEW**

2. تظهر لك نافذة تختار منها الاختيار الأول TEXT ثم تقوم بالضغط على موافق

3. تأخذ مسافة قيمتها تاب واحد بالضغط على المفتاح TAB و قبلها تضع في بداية السطر الحرف C

4. تكتب البرنامج التالي :

```
SOLUTION  C     THIS PROGRAM TO CALCOLATE
METHOD OF EQUATION BY USED BISECTION
RAED ( *,* ) A , B
P=(A +B ) / 2
Y1= A^3 + 4*A^2 – 10
Y2= B^3 + 4*B^2 -10
Y0=P^3 + 4*P^2 – 10
IF  ( ASB( Y0) .LT. 0.00001  )  GOTO  100
IF  (Y0*Y1 .GT. 0.0 ) GOTO  10
A=A
B=B
GOTO 20
A=-P
B=B
GOTO 20
WRITE(*,*)   Y0 , P
STOP
END
```

5. تحفظ البرنامج باسم BISECTION.FOR ثم و هذه الخطوة مهمة جدا حيث يتحول البرنامج من مجرد ملف نصي TEXTإلى لغة FORTRAN و تظهر علامات معالج اللغة أي الكلمات المحجوزة للغة بخط ازرق و خط اخضر يحتوي البرنامج كله دليل على أن هذه الكلمات تحتوي على نص بلغة الفورتران

6. من القائمة المنسدلة BUILDاختر COMPILE BISECTION.FOR

7. تظهر لك نافذة في أسفل البرنامج إن كان البرنامج يحتوي على أخطاء فتبين لك الأخطاء و عليك المراجعة و التحقق و إلا فتكون النتيجة هي : (0 WARING(S) 0 ERROR(S

8. من القائمة المنسدلة BUILD اختر BUILD BISECTION.EXE

9. من نفس القائمة المنسدلة السابقة اختر الأمر EXECUTE BISECTION.EXE

10. تظهر لك نافذة ضع بها القيمة 1 و اضغط ENTER والقيمة 2 ثم اضغط ENTER

2-2 طريقة نيوتن Nuten method

في هذا الدرس نستعرض سويا احد الطرق المتبعة في التحليل العددي بعد أن تكلمنا عن مفهوم التحليل العددي و أهميته و الطريقة الثانية المتبعة في التحليلي العددي هي طريقة نيوتن (newten method) و لا أود أن أتطرق إلى الاستنتاج الرياضي لهذه الطريقة بالقدر الذي ارغب فيه في التركيز على البرامج الرياضية لها, تعتبر هذه الطريقة من الطرق السهلة في إيجاد القيم التقريبية للمعدلات الرياضية و الاختلاف الأساسي بيناه و بين الطريقة السابقة (طريقة المقاطع) (bisection method) هو سهولة استخدام طريقة نيوتن و سرعتها في إيجاد الحل التقريبي للمعادلة.

المفهوم الرياضي لنيوتن

اذا كانت لدينا دالة حقيقية f(x) و كان x0 هو الجدر للمعادلة المطلوب الحصول عليه (الحل التقريبي للمعادلة) فيمكن إيجاده عن طريق الآتي:

X2=x1*f(x)/f '(x)

حيث :

X2 هي القيمة التي نبحث عنها أي نفس قيمة الجدر (قيمة x0)

X1 هي القيمة المدخلة عند القراء

F(x) هي قيمة المعادلة بعد تعويض بي قيمة x1 في المعادلة

F ' (x) هي قيمة المشتقة الأولى للمعادلة بعد.

و تكتب خوارزمية البرنامج كالأتي:

- 1. Read x1
- 2. f(x1)=x1^3+4*x1^2-10

- 3. f ' (x1) = 3*x1^2 +8*x1
- 4. if f(x1) > 0.00001 write x2,f(x1)
- 5. x2=x1-(f(x1)/ f ' (x1) ,x1=x2 goto step (2)
- 6. write x2 ,p
- 7. stop

شرح الخوارزمية

يبدأ البرنامج بقراءة قيمة الدالة عند النقطة (x1) ثم تتم عملية تعويض من اجل إيجاد القيمة الفعلية للدالة عند نفس النقطة، تعوض قيمة (x1) في مشتقة الدالة و كأننا أوجدنا الميل في هذه الحالة، تتم اختبار قيمة الدالة f (x1) فإذا كانت أصغر من 0.00001 فتتم طباعة القيمة مباشرة و إلا ننتج قيمة جديدة هي x2 من طرح قيمة x1 من مقسوم الدالتين f (x) , f ' (x) و هي قيمة أو نقطة تقاطع المماس مع محور السينات و عندما نجعل قيمة x1=x2 فإننا نقترب من الحل أكثر أي من النقطة الصفرية التي تحقق الحل.

و هذا هو البرنامج بلغة FORTRAN:-

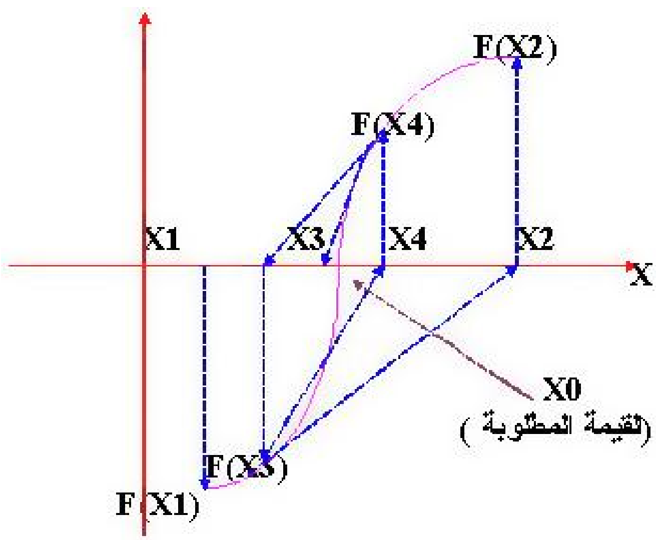
```
   Read(*,*) x1
10 f(x1)=x1^3+4*x1^2-10
   f ' (x1) = 3*x1^2 +8*x1
   if f(x1) > 0.00001) goto 20
   x2=x1-(f(x)/f ' (x))
   x1=x2
   goto 10
20 Write(*,*) x2 ,p
   stop
   end
```

و يمكن استخدام جملة DO بدلا من جملة GOTO لأداء نفس المهمة على النحو الآتي:

```
Read(*,*) x1
DO 500,I=1,N
f(x1)=x1^3+4*x1^2-10
```

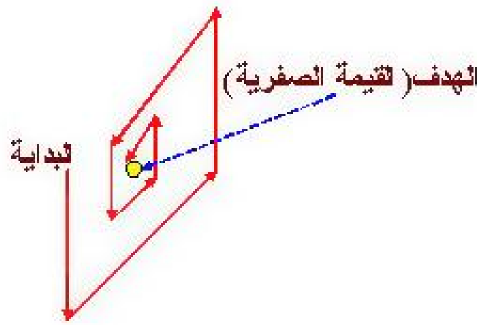
- f ' (x1) = 3*x1^2 +8*x1
- if f(x1) > 0.00001)   GOTO 20
- x2=x1-(f(x)/f ' (x))
- x1=x2
- GOTO 500
- 500   CONTINUE
- 20   Write(*,*) x2 ,p
- stop
- end

و الرسم التالي يوضح فكرة نيوتن في حل المعادلات :



(شكل2-2)

و نلاحظ انه بدأ من نقطة بعيدة و في كل مرة يقترب بمقدار معين من القيمة المطلوبة X0 و كأنه يأخذ الشكل التالي:



(شكل 2-3)

2-3 طريقة نيوتن رافسن ونيوتن رافسن المعدلة

Newton-Raphson Method
And
Modified Newton-Raphson Method

طريقتا نيوتن رافسن ونيوتن رافسن المعدلة:

هما طريقتان لحل المعادلات الآنية غير الخطية , ولو أخذنا معادلتين في متغيرين كمثال ونريد ايجاد قيمتي x,y اللتان تحققان المعادلة (1)

$$F1(x,y)=0 \qquad\qquad F2(x,y)=0 \ldots\ldots\ldots\ldots(1)$$

نستخدم لذلك إحدى الطريقتين

أولا: طريقة نيوتن رافسن

من متسلسلة تايلور في متغيرين كمثال

$$F1(x_{i+1},y_{i+1})=F1(x_i,y_i)+\frac{\partial F1}{\partial x}(x_{i+1}-x_i)+\frac{\partial F1}{\partial y}(y_{i+1}-y_i)+\ldots\ldots.$$

$$F2(x_{i+1},y_{i+1})=F2(x_i,y_i)+\frac{\partial F2}{\partial x}(x_{i+1}-x_i)+\frac{\partial F2}{\partial y}(y_{i+1}-y_i)+\ldots\ldots.$$

نفترض أن xi+1,yi+1 قريبة جدا من الحل لذلك

$$F1(xi+1,yi+1)=F2(xi+1,yi+1)=0$$

ونضع كذلك

$$x_{i+1} - x_i = h$$
$$y_{i+1} - y_i = k$$

لحصلنا على الصورة

$$0 = F1(x_i,y_i)+\frac{\partial F1}{\partial x}h+\frac{\partial F1}{\partial y}k$$

$$0 = F2(x_i,y_i)+\frac{\partial F2}{\partial x}h+\frac{\partial F2}{\partial y}k$$

ومنها

$$\frac{\partial F1}{\partial x}h+\frac{\partial F1}{\partial y}k = - F1(x_i,y_i)$$

$$\frac{\partial F2}{\partial x}h+\frac{\partial F2}{\partial y}k=-F2(x_i,y_i)$$

وهذه المعادلة غير متجانسة ونحصل على حل عندما محدد Jacobian لايساوي الصفر

$$J=\begin{vmatrix}\partial F1/\partial x & \partial F1/\partial x \\ \partial F2/\partial y & \partial F2/\partial y\end{vmatrix}\neq 0$$

ونحصل على المعاملات كالتالي:

$$H=\begin{vmatrix}-F1(xi,yi) & \partial F1/\partial y \\ -F2(xi,yi) & \partial F2/\partial y\end{vmatrix}/J$$

$$K=\begin{vmatrix}\partial F1/\partial x & -F1(xi,yi) \\ \partial F2/\partial x & -F2(xi,yi)\end{vmatrix}/J$$

نستخدم المعادلتين التاليتين للحصول على قيمتين جديدتين لكل من x,y غير الافتراضيتين

$$X_{i+1}=xi+h$$

$$Y_{i+1}=yi+k$$

نعيد الكرّة من جديد باستخدام القيمتين الجديدتين .

ثانيا: طريقة نيوتن رافسن المعدلة

من الواضح صعوبة الطريقة السابقة في حالة تزايد عدد المعادلات (n) أي سيكون لدينا (n^2) من المشتقات الجزئية لكل معادلة n من المشتقات والمتغيرات , لذلك تُعدل الطريقة كالآتي: نحصل على قيمة x الجديدة من طرح الحالية من إحدى الدالتين على تفاضلها الجزئي بالنسبة ل x أما قيمة y الجديدة فمن طرح الحالية من الدالة الأخرى على تفاضلها الجزئي بالنسبة ل y .
واختيار أي الدالتين نستخدمه مع x أو y ليس عشوائيا إنما يكون الاختيار

مهماً حيث يؤدي الاختيار الصحيح إلى تقارب نحو الحل أما الخاطئ فسوف يبعدك عن الحل. وهنا كمثال اخترنا F1 مع x و F2 مع y

$$x_{i+1}=x_i-\frac{F_1(xi,yi)}{\frac{\partial F_1(xi,yi)}{\partial x}} \quad \text{..........................(1)}$$

$$y_{i+1}=y_i-\frac{F_1(xi,yi)}{\frac{\partial F_1(xi,yi)}{\partial x}} \quad \text{..............................(2)}$$

بالنسبة للدالتين : $y=x+\sin(x)$ و $3\sin(x)-y=0$

واللتان يُراد نقاط الحل لهما أي نقاط تقاطعهما ويبين الرسم في الصفحة التالية نقاط التقاطع. وسنقوم بايجاد الحل بالطريقتين:

بطريقة نيوتن رافسن

$$F1(x,y)=x+\sin(x)-y$$
$$F2(x,y)=3\sin(x)-y$$

نفاضل كلا الدالتين بالنسبة لx وبالنسبة لy

$$\frac{\partial F1}{\partial x}=1+\cos(x) \qquad \frac{\partial F1}{\partial y}=-1$$

$$\frac{\partial F2}{\partial x}=3\cos(x) \qquad \frac{\partial F2}{\partial y}=-1$$

$$J=\begin{vmatrix} 1+\cos(x) & -1 \\ 3\cos(x) & -1 \end{vmatrix} = -(1+\cos(x))+3\cos(x)$$

$$= 2\cos(x)-1 \quad \text{..............................(i)}$$

$$H=\begin{vmatrix} -(x+\sin(x)-y) & -1 \\ -(3\sin(x)-y) & -1 \end{vmatrix}/J = [(x+\sin(x)-y)-(3\sin(x)-y)]/J$$

$$= (x-2\sin(x))/J \quad \text{............................(ii)}$$

$$K=\begin{vmatrix} 1+\cos(x) & -(x+\sin(x)-y) \\ 3\cos(x) & -(3\sin(x)-y) \end{vmatrix}/J =$$

$$=(-(3\sin(x)-y)(1+\cos(x))+3\cos(x)(x+\sin(x)-y))/J$$
$$=(-3\sin(x)+y-3\sin(x)\cos(x)+y\cos(x)+3x\cos(x)+3\sin(x)\cos(x)-3y\cos(x))/J$$
$$=(-3\sin(x)+y-2y\cos(x)+3x\cos(x))/J$$
$$=(y-3\sin(x)+\cos(x)(3x-2y))/J \quad \ldots\ldots\ldots\ldots\ldots\ldots(iii)$$

قائمة بالمتغيرات المستخدمة في الخوارزمية والبرنامج

I: متغير للتكرار

X, y : النقطة الحالية

Xnew, ynew: تمثل النقطة الجديدة

Jac: معامل جاكوبي ويساوي المعادلة (**i**)

H: معامل تفاضل الدالة بالنسبة لـ**x** ويساوي المعادلة (**ii**)

k: معامل تفاضل الدالة بالنسبة لـ**y** ويساوي المعادلة (**iii**)

E: مقدار الخطأ المسموح به وقد اعتبرته **0.00001**

الخوارزمية: (نيوتن رافسن)

1. أبدا
2. ادخل قيمة x , y الأولية.
3. اجعل I=0 وهي تمثل عدد التكرارات
4. اطبع قيمة I , x , y الأولية.
5. اجعل المتغير jac=2*cos(x)-1
6. هل jac=0 إذا كان نعم.... اطبع القسمة على الصفر ممنوعة ثم اذهب إلى 13
7. اجعل h=(x-2*sin(x))/jac و k=(y-3*sin(x)+cos(x)(3*x-2*y))/jac
8. اجعل xnew=x+h و ynew=y+k
9. أضف 1 إلى i
10. اطبع قيم I , x, y الجديدة

11. هل القيمة المطلقة للفرق بين x الحالية والجديدة أقل من E إذا كان

لا..... استبدل x الحالية بالجديدة انتقل إلى 5

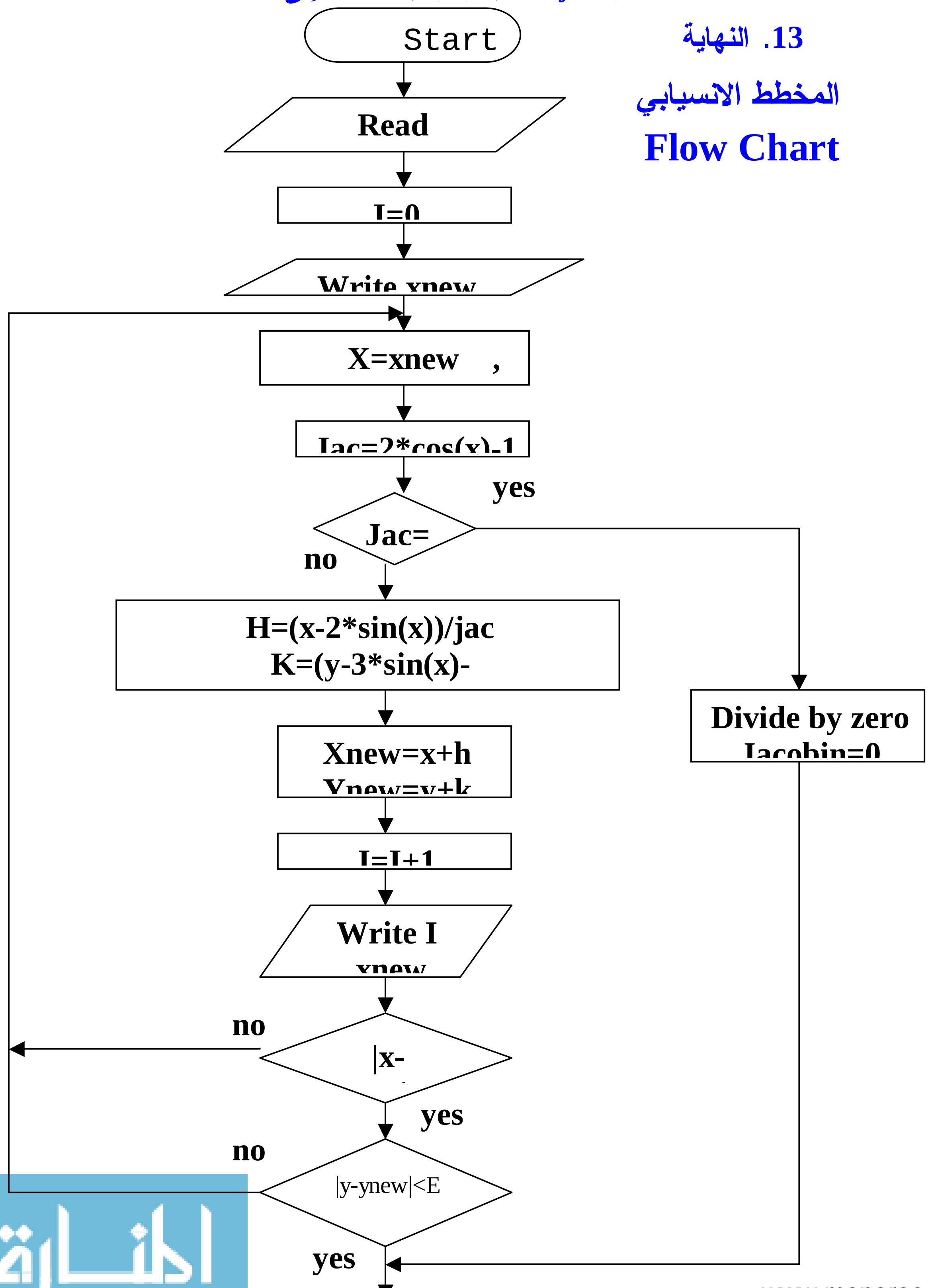
12. هل القيمة المطلقة للفرق بين y الحالية و الجديدة أقل من E إذا كان

لا..... استبدل y الحالية بالجديدة انتقل إلى 5

13. النهاية

المخطط الانسيابي

Flow Chart



شكل (2-4)

البرنامج

```
# include<conio.h>
# include<stdio.h>
# include<math.h>
void main()
{  FILE *stream;
 int i;
 float x,y,xnew,ynew,jac,h,k;
 printf("Enter initial value of x, y  :");
 scanf("%f %f",&xnew,&ynew);
 i=0;
 stream = fopen("newraf.FIL", "w+");
 fprintf(stream,"The initial value of x,y is (%2.1f
 ,%2.1f ) \n",xnew,ynew);
 fprintf(stream,"\n i\t x\t\t y\t");
          fprintf(stream,"\n_________________________________________");

 fprintf(stream,"\n %d\t %f\t %f\t",i,xnew,ynew);
 do
 { x=xnew;
   y=ynew;
   jac=2*cos(x)-1 ;
   if( jac ==0) {
     fprintf(stream," divide by zero , Jacobin=0  ");
     break; }
   h=(x-2*sin(x))/jac;
   k= (y-3*sin(x)-2*y*cos(x)+3*x*cos(x))/jac;
   xnew=x+h;
   ynew=y+k;
   i++;
   fprintf(stream,"\n %d\t %f\t %f\t",i,xnew,ynew);
 }while(fabs(x-xnew)>.00001 & fabs(y-ynew)>.00001);
fclose(stream);
}
```

الإدخالات و النتائج

مــن الرسم يتبين أن هناك ثلاثة جذور أحدهم (0,0) والآخران أحدهما موجب والآخر سالب لكل من x,y ومن هنا سنحدد قيم x,y الابتدائيات

```
Enter initial value of x, y  :2 1

The initial value of x,y is (2.0,1.0)

 i   x       y
__________________________________
0   2.000000   1.000000
1   1.900996   2.851493
2   1.895512   2.843267
3   1.895494   2.843241
4   1.895494   2.843241
```

الجذر الثاني

```
Enter initial value of x, y  :-2 -1

The initial value of x,y is (-2.0,-1.0)

 i   x       y
__________________________________
0   -2.000000  -1.000000
1   -1.900996  -2.851493
2   -1.895512  -2.843267
3   -1.895494  -2.843241
4   -1.895494  -2.843241
```

الجذر الثالث

```
Enter initial value of x, y  :0.5 0.5

The intial value of x,y is (0.5,0.5)

 i   x       y
__________________________________
0   0.500000   0.500000
1   -0.107617  -0.161425
2   0.000840   0.001259
3   -0.000000  -0.000000
4   0.000000   0.000000
```

أما الطريقة المعدلة

لو أخذنا الدالتين بهذا الترتيب

F1 (x,y)=3sin(x)-y
F2 (x,y)=x+sin(x)-y

ونأخذ للأولى تفاضل بالنسبة ل x والثانية بالنسبة ل y

$\frac{\partial F1}{\partial x}$ =3cos(x) $\frac{\partial F2}{\partial y}$ = -1

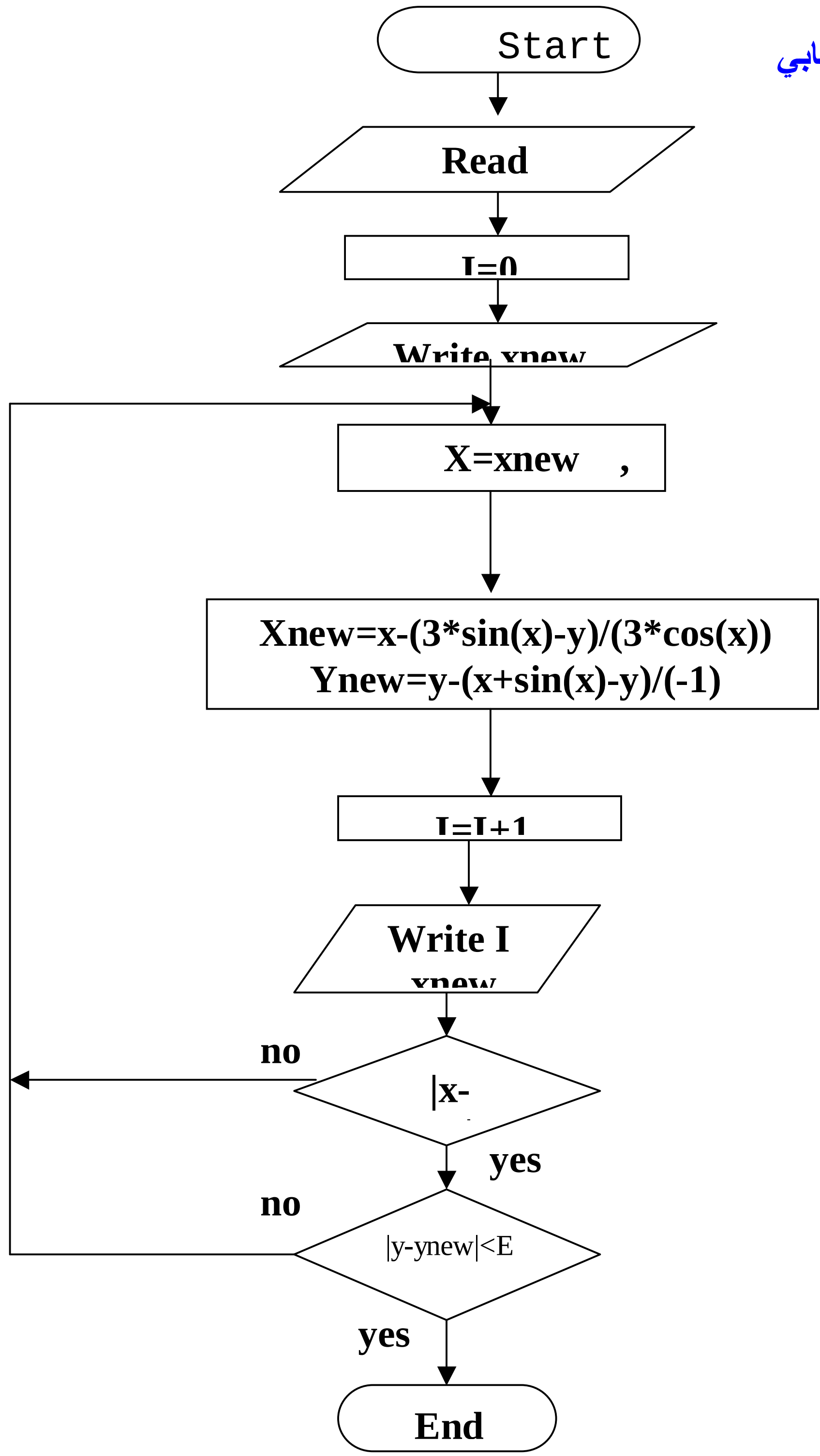
ونطبق القانون

X_{i+1} = xi – (3sin(xi)-yi)/ (3cos(xi))
Y_{i+1} = yi – (xi+sin(xi)-yi)/(-1)

الخوارزمية:

1. أبدا
2. ادخل قيمة x , y الأولية.
3. اجعل I=0 وهي تمثل عدد التكرارات
4. اطبع قيمة I , x , y الأولية.
5. اجعل xnew=x-(3*sin(x)-y)/(3*cos(x)) و ynew=y-(x+sin(x)-y)/(-1)
6. أضف 1 إلى i
7. اطبع قيم I , xnew, ynew التي تمثل قيم I,x,y الجديدة
8. هل القيمة المطلقة للفرق بين x الحالية والجديدة أقل من E إذا كان
لا..... استبدل x الحالية بالجديدة (xnew) ثم انتقل إلى 5
9. هل القيمة المطلقة للفرق بين y الحالية والجديدة أقل من E إذا كان
لا..... استبدل y الحالية بالجديدة (ynew) ثم انتقل إلى 5
10. النهاية

المخطط الانسيابي
flow chart



(شكل 2-5)

البرنامج

```
#include<conio.h>
# include<stdio.h>
# include<math.h>
void main()
{  FILE *stream;
int i;
float x,y,xnew,ynew;
 printf("Enter initial value of x, y");
 scanf("%f %f",&xnew,&ynew);
  i=0;
 stream = fopen("modnera.FIL", "w+");
fprintf(stream,"The initial value of x,y is
(%2.1f ,%2.1f)\n",xnew,ynew);
fprintf(stream,"\n i\t x\t\t y\t");
fprintf(stream,"\n_____________________________________________");

fprintf(stream,"\n %d   %f
%f",i,xnew,ynew);
do
{ x=xnew;
  y=ynew;
 xnew=x-(3*sin(x)-y)/(3*cos(x));
 ynew=y-(x+sin(x)-y)/(-1);
 i++;
fprintf(stream,"\n %d   %f
%f",i,xnew,ynew);
}while(fabs(x-xnew)>.00001 & fabs(y-
ynew)>.00001);
fclose(stream);
}
```

وبإعطائها القيم

```
Enter initial value of x, y  :2 1

The initial value of x,y is (2.0,1.0)

i      x          y
```

i	x	y
0	2.000000	1.000000
1	3.384041	2.909297
2	2.137745	3.143961
3	1.757074	2.981289
4	1.697342	2.739774
5	2.321275	2.689346
6	2.079209	3.052638
7	1.783338	2.952727
8	1.751365	2.760836
9	2.104744	2.735107
10	2.004736	2.965549
11	1.811609	2.912052
12	1.813539	2.782754
13	1.992840	2.784222
14	1.954219	2.905093
15	1.844689	2.881608
16	1.852782	2.807414
17	1.941547	2.813287
18	1.925796	2.873602
19	1.867622	2.863443
20	1.873738	2.823891
21	1.917870	2.828201
22	1.910927	2.858242
23	1.880843	2.853638
24	1.884470	2.833162

25	1.906570	2.835676
26	1.903323	2.850726
27	1.887941	2.848544
28	1.889914	2.838070
29	1.901036	2.839426
30	1.899461	2.847000
31	1.891635	2.845935
32	1.892669	2.840606
33	1.898283	2.841314
34	1.897503	2.845137
35	1.893532	2.844607
36	1.894064	2.841903
37	1.896901	2.842267
38	1.896511	2.844199
39	1.894499	2.843934
40	1.894770	2.842563
41	1.896206	2.842748
42	1.896009	2.843725
43	1.894990	2.843592
44	1.895128	2.842898
45	1.895854	2.842992
46	1.895755	2.843486
47	1.895239	2.843419
48	1.895309	2.843067
49	1.895676	2.843115
50	1.895626	2.843365
51	1.895365	2.843331
52	1.895400	2.843153
53	1.895586	2.843177
54	1.895561	2.843304
55	1.895429	2.843287
56	1.895447	2.843197
57	1.895541	2.843209
58	1.895528	2.843273
59	1.895461	2.843264

(جدول 2-1)

```
Enter initial value of x, y  :0.5 0.5

The initial value of x,y is (0.5,0.5)

i     x           y
________________________________________

 0    0.500000    0.500000
 1    0.143613    0.979426
 2    0.328876    0.286733
 3    0.088597    0.651855
 4    0.217908    0.177078
 5    0.056940    0.434095
 6    0.144872    0.113849
 7    0.037329    0.289237
 8   0.074650     0.264690
9    0.024699     0.192775
10   0.064273    0.049396
11   0.016411    0.128502
12   0.042838    0.032821
13   0.010924    0.085663
14   0.028556    0.021848
15   0.007278    0.057107
16   0.019036    0.014556
17   0.004850    0.038071
18   0.012691    0.009701
19   0.003233    0.025381
20   0.008460    0.006466
21   0.002155    0.016920
22   0.005640    0.004311
       0.001437                           0.011280
 2
 3
24   0.003760    0.002874
25   0.000958    0.007520
26   0.002507    0.001916
27   0.000639    0.005013
28   0.001671    0.001277
29   0.000426    0.003342
```

```
30   0.001114    0.000851
31   0.000284    0.002228
32   0.000743    0.000568
33   0.000189    0.001485
34   0.000495    0.000378
35   0.000126    0.000990
36   0.000330    0.000252
37   0.000084    0.000660
38   0.000220    0.000168
39   0.000056    0.000440
40   0.000147    0.000112
41   0.000037    0.000293
42   0.000098    0.000075
43   0.000025    0.000196
44   0.000065    0.000050
45   0.000017    0.000130
46   0.000043    0.000033
47   0.000011    0.000087
48   0.000029    0.000022
49   0.000007    0.000058
50   0.000019    0.000015
51   0.000005    0.000039
52   0.000013    0.000010
```

(جدل 2-2)

```
Enter initial value of x, y  :-2 -1

The initial value of x,y is (-2.0,-1.0)

i     x          y
______________________________________

 0   -2.000000   -1.000000
 1   -3.384041   -2.909297
 2   -2.137745   -3.143961
 3   -1.757074   -2.981289
 4   -1.697342   -2.739774
 5   -2.321275   -2.689346
 .       .            .
 .       .            .
57  -1.895541    -2.843209
58  -1.895528    -2.843273
```

```
59   -1.895461    -2.843264
```

وبتغيير بسيط في البرنامج بحيث نستغل قيمة x الجديدة التي تم الحصول عليها من المعادلة الأولى واستخدامـــــــها في المعادلة الثانية للحصول على y الجديدة أي y من (x_{i+1},yi) بدلا من الطريقة السابقة والتي نحصل فيها على y من (xi,yi)

$X_{i+1} = xi – (3\sin(x_i)\text{-}yi\ )/\ (3*\cos(xi))$
$Y_{i+1} = yi – (x_{\ i+1}+\sin(x_{i+1})\text{-}yi)/(\text{-}1)$

وندرجها في البرنامج كالتالي:

```
.......................
xnew=x-(3*sin(x)-y)/(3*cos(x));
 ynew=y-(xnew+sin(xnew)-y)/(-1);
.......................
```

فكانت النتائج لنفس الإدخالات كالتالي:

Enter initial value of x, y :2 1

```
The initial value of x,y is (2.0,1.0)
```

i	x	y
0	2.000000	1.000000
1	3.384041	3.143961
2	2.057167	2.941202
3	1.851002	2.812001
4	1.936573	2.870419
5	1.872380	2.827247
6	1.914300	2.855880
7	1.883480	2.834992
8	1.904657	2.849441
9	1.889309	2.839011
10	1.900055	2.846337
11	1.892333	2.841084
12	1.897785	2.844799
13	1.893886	2.842145

14	1.896649	2.844028
15	1.894678	2.842685
16	1.896078	2.843639
17	1.895080	2.842959
18	1.895790	2.843442
19	1.895285	2.843099
20	1.895644	2.843343
21	1.895388	2.843169
22	1.895570	2.843293
23	1.895441	2.843205
24	1.895532	2.843267
25	1.895467	2.843223
26	1.895514	2.843255
27	1.895481	2.843232
28	1.895504	2.843248
29	1.895487	2.843237
30	1.895499	2.843245

(جدول 3-2)

Enter initial value of x, y :0.5 0.5

The initial value of x,y is (0.5,0.5)

I	x	y
0	0.500000	0.500000
1	0.143613	0.286733
2	0.095576	0.191007
3	0.063669	0.127295
4	0.042432	0.084850
5	0.028283	0.056563
6	0.018854	0.037708
7	0.012569	0.025138
8	0.008379	0.016759
9	0.005586	0.011172
10	0.003724	0.007448
11	0.002483	0.004966
12	0.001655	0.003310
13	0.001103	0.002207

14	0.000736	0.001471
15	0.000490	0.000981
16	0.000327	0.000654
17	0.000218	0.000436
18	0.000145	0.000291
19	0.000097	0.000194
20	0.000065	0.000129
21	0.000043	0.000086
22	0.000029	0.000057
23	0.000019	0.000038

(جدول2-4)

و بتغيير اختيار الدوال(إعادة الترتيب) كالآتي:

$$F1(x,y)= x+\sin(x)-y \quad , \quad F2(x,y)= 3\sin(x)-y$$

ونأخذ للأولى تفاضل بالنسبة ل x والثانية بالنسبة ل y

$$\frac{\partial F1}{\partial x}=1+\cos(x) \quad , \quad \frac{\partial F2}{\partial y}= -1$$

ونطبق القانون

$$X_{i+1} = xi - (xi+\sin(xi)- yi)/ (1+\cos(xi))$$
$$Y_{i+1} = yi - (3\sin(xi)-yi)/(-1)$$

فإن الناتج سيكون مبتعداً عن الحل وهذا هو ما حصل فعلاً

0	3.000000	2.000000
1	-111.026443	0.423360

....

71	-1431034113556480.000000	-1.920399
72	49127493789024256.000000	-0.708711
73	+NAN	+NAN

الاستنتاج

* فـــي الطريقة المعدلة وبعد استخدام قيمة x الجديدة لإيجاد y الجديدة تقلص عدد التكرار إلى النصف تقريبا عن الطريقة المعدلة التي استخدمت فيها قيمة x القديمة لإيجاد y الجديدة أي وجدنا الحل في نصف الوقت.

* فـــي طـــريقة نـــيوتن رافـــسون تم الوصول إلى الحل بأقل تكرار من المعدلة وبـــتقارب دائم نحو الحل بينما الطريقة المعدلة نجدها مرة تتقارب وأخرى تبتعد مما يزيد عدد التكرار. وهذا يتضح من الأشكال التوضيحية لاحقاً.

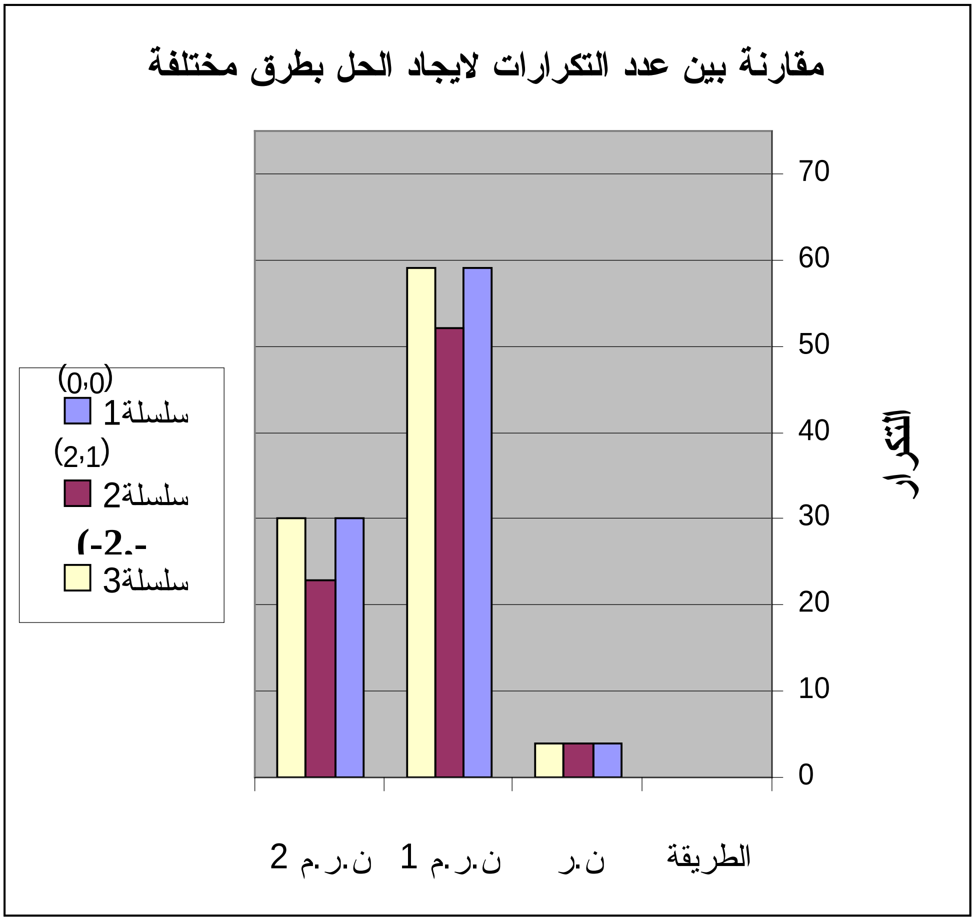
* اختيار أي معادلة نعتبرها F1(x,y) مهم. حيث نلاحظ التقارب والتباعد الناتج عن ذلك

* نلاحظ من الجدول, اختلاف نتائج الطرق سالفة الذكر من حيث عدد التكرارات حسب قيمة x,y المدخلة مع ملاحظة أي من الحلول الثلاثة نصل إليه في كل مرة. ومن المعلوم أن هذه التكرارات تعتمد على E (قيمة الخطأ المسموح به) .

نيوتن رافسن المعدلة مع استغلال x الجديدة لإيجاد y	نيوتن رافسن المعدلة	نيوتن رافسن	النقطة المدخلة
التكرار > الحل	التكرار > الحل	التكرار > الحل	
(0,0) > 24	(-1.8,-2.8) > 61	(-1.8,-2.8)>5	-3,-5
(0,0) > 24	(0,0) > 54	(0,0) > 5	0.6,0.6
(-1.8,-2.8)>33	(-1.8,-2.8)>75	(1.8,2.8)>5	1.5,0.75
(0,0)>27	تباعد	(1.8,2.8)>5	1.5,2.5
(1.8,2.8)>29	(1.8,2.8)>57	(1.8,2.8)>4	2,2
(0,0)>24	(1.8,2.8)>61	(1.8,2.8)>5	3,5
(0,0)>46	تباعد	(1.8,2.8)>114	7,4
(0,0)>30	تباعد	(-1.8,-2.8)>7	10,12
(0,0)>31	تباعد	(1.8,2.8)>145	20,20

(جدول 2-5)

شكل توضيحي يبين مقارنة بين الطرق من حيث التكرارات



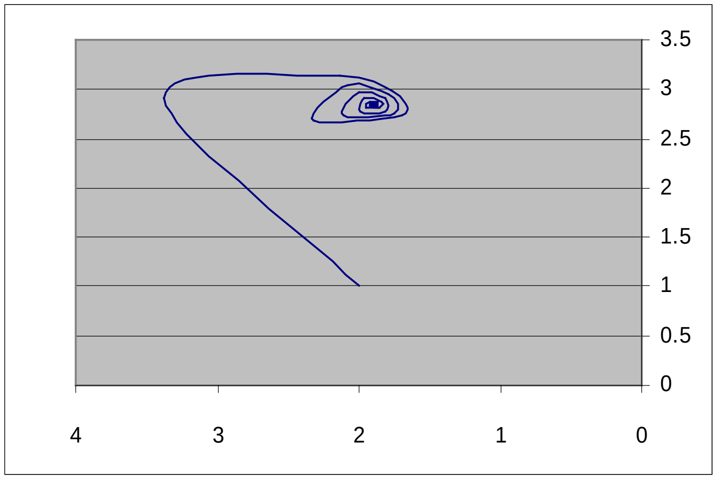
(شكل 2-6)

ن.ر : طريقة نيوتن رافسون

ن.ر.م 1 : طريقة نيوتن رافسون المعدلة

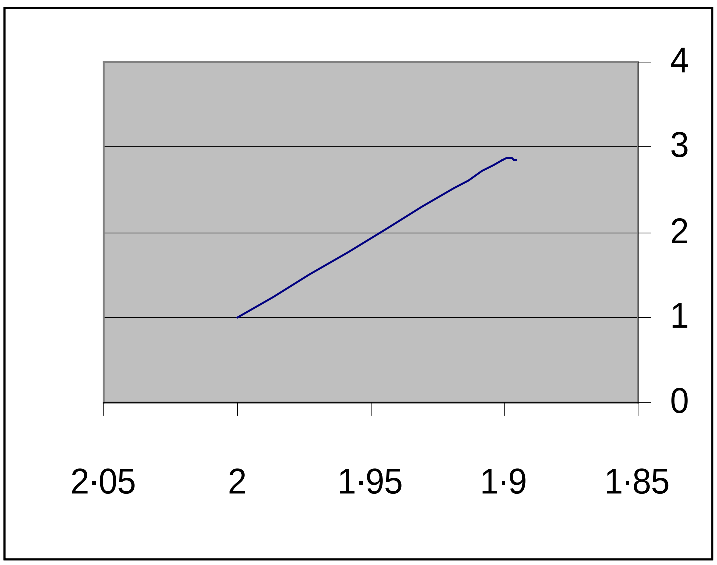
ن.ر.م 2 طريقة نيوتن رافسون المعدلة والتي قمنا فيها باستغلال قيم x الجديدة لايجاد y

شـــكل يوضح تقارب الـــنقطة (2,1) إلى الحــل فـــي طريقة نيوتن رافسن



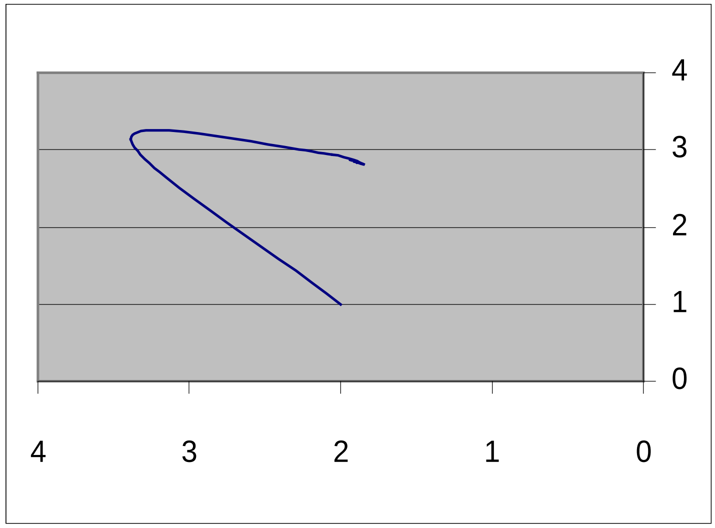
(شكل2 - 7)

شـــكل يوضـــح تقارب النقطة (2,1) إلى الحل في طريقة نيوتن رافسن المعدلة



(شكل 2 - 8)

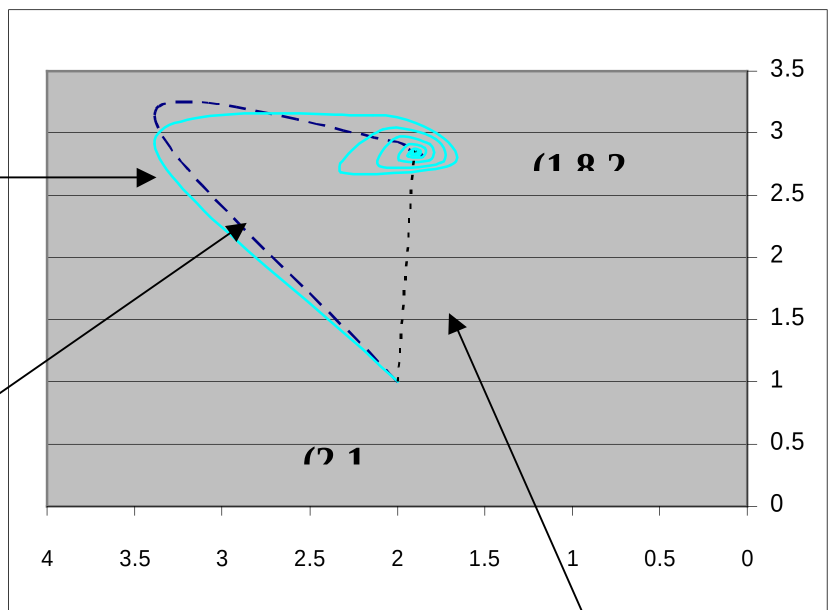
شــكل يوضح تقارب الــنقطة (2,1) إلـــى الحــل فـــي طريقة نيوتن رافسن المعدلة باســـتعمال قـــيم **x** الجديدة لإيجاد **y**



(شكل 2-9)

الأشكال الثلاثة في شكل واحد مع بيان نقطة البدء ونقطة الحل

خـــط يوضح تقارب الـــنقطة (2,1) إلـــى الحـــل فـــي طريقة نيوتن رافسن المعدلة باســـتعمال قـــيم **x** الجديدة لإيجاد **y**



خط يوضح تقارب النقطة (2,1) إلى الحل في طريقة نيوتن رافسن المعدلة

(شكل 2-10)

خـــط يوضح تقارب الـــنقطة (2,1) إلى الحـــل فـــي طريقة نيوتن رافسن

4-2 طريقة الموقع الخاطئ

False position

كنت قد شرحت طريقتين من طرق حل المعادلات ذات المجهول الواحد، و نظرا لأهمية هذه الطرق ، رأيت أن أضيف هذه الطريقة، و التي لا تقل أهمية عن الطرق السابقة ، وللمزيد من الاتساع في الموضوع و لان التعدد يفتح آفاق الخيار أمام الدارس من اجل التنوع و الاستفادة الشاملة، كنت أريد أن ابدأ في شرح (Lagrange interpolation polynomial) لا اعرف ترجمة عربية دقيقة لها، على كل حال لنبدأ في طريقة (الموقع الخاطئ) (false position)

المفهوم النظري

إذا كانت الدالة معرفة في الفترة (x1 , x2) و كان f(x1)*f(x2)<0

من الرسم الموضع يمكن استنتاج أن المثلثين :

0 , f(x1) , f(x2)

x3 , x2 , f(x2)

متشابهان فيكون :

F(x2)/{f(x2) – f(x1)}=(x2 – x1)/ (x2 – x1)

و تكون قيمة x3

X3=x2 – f(x2)*{(x2- x1)/f(x2)-f(x2)}

و يمكن كتابة الخوارزمية للمعادلة التالية

y1=x**3 +4*x**2-10

مع ملاحظة أن الرمز ** يعني في لغة الفورتران للقوة الثانية مثلا أو الثالثة أو غيرها أي الأس

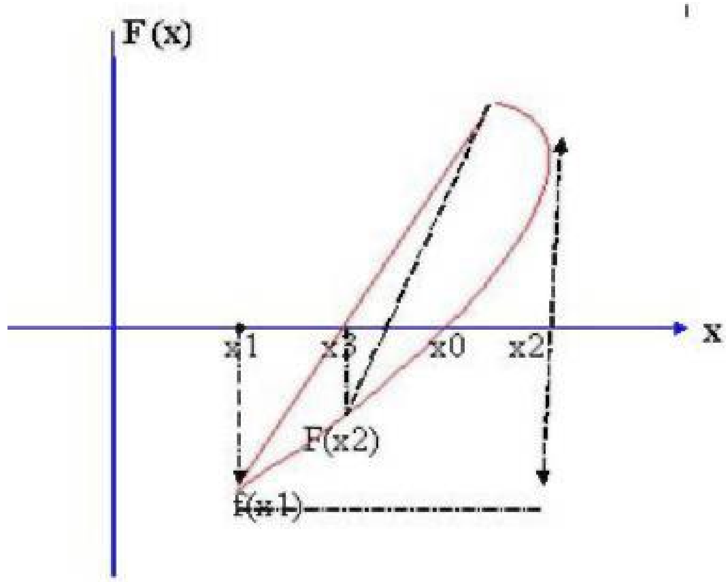
كالأتي:

- start .1
- read(x1 ,x2) .2

- 3. do
- y1=x1**3 +4*x1**2-10
- y2=x2**3+4*x2**2-10
- X3=x2 – f(x2)*((x2- x1)/f(x2)-f(x2))
- Y3=x3**2+4*x3**2-10
- 4. if(y3<0.000001)goto (8)
- 5. if(y1*y3>0.0)goto (7)
- 6. x1=x1
- x3=x2
- goto (3)
- 7. x1=x3
- x2=x2
- goto 3
- 8. write x3 ,y3
- 9. stop

و البرنامج بلغة فورتران كالأتي :

- Read(*,*)x1 ,x2
- y1=x1**3 +4*x1**2-10
- y2=x2**3+4*x2**2-10
- X3=x2 – f(x2)*((x2- x1)/f(x2)-f(x2))
- Y3=x3**2+4*x3**2-10
- if(abs(y3).lt.0.000001)goto 8
- if(y1*y3.gt.0.0)goto
- x1=x1
- x3=x2
- goto 3
- x1=x3
- x2=x2
- goto 3
- write(*,*) x3 ,y3
- stop
- End



(شكل 2-11)

و بهذا نكون قد انهينا طريقة الموقع الخاطئ

الباب الثالث

الحلول العددية للمعادلات التفاضلية العادية

3-1 الحلول العددية للمعادلات التفاضلية العادية

سوف نقوم بحل المعادلات من النوع

dy/dx = f(x,y)

أي معادلـــة خطـــية من الرتبة الأولى . ولحل هذا النوع من المعادلات هناك مجموعة من الطرق منها:

3-2 طريقة اويلر Euler's Method

نفترض أن الحل **y=F(x)** متصل وقابل للتفاضل وأن نقطة البداية هي **(x0,y0)** وأن الحل مطلوبا عند **x=x\`** لذلك يجب تجزئة الفترة إلى **n** من الاجزاء بعرض **w** (على أن تكون صغيرة)

حيث **w = (x\`-x)/n**

ثم نطبق القانون

$$y_{i+1}=yi +w * f(xi,yi)$$

حيث **I = 0,1,..n-1**

ملاحظة:

- يكون الخطأ أقل بقيم أقل لـ**w**
- تزداد الحسابات بازدياد عدد الفترات أي تصغير **w**
- كل **y** تعتمد على التي قبلها مما ينشر الخطأ في حالة وقوعه.

امتداد طريقة اويلر Extended Euler Method

بأخذ ثلاثة حدود من متسلسلة تايلور التي حصلنا منها على **EM** نحصل على **EEM**

$$y_{i+1}=yi +w * f(xi,yi) + w^2/2! * f\,`(xi,yi)$$

حيث **I = 0,1,..n-1**

3-3 طريقة اويلر الاكثر امتدادا More Extended Euler Method

بأخذ الحد الرابع من متسلسلة تايلور نحصل على **MEEM**

$$y_{i+1}=yi +w * f(xi,yi) + w^2/2! * f\,`(xi,yi) + w^3/3! * f\,``(xi,yi)$$

3-4 طريقة اويلر المعدلة Modified Euler Method

في هذه الطريقة نستغل **EM** لإيجاد **y`i+1 (Predictor)** ثم نستخدمها لحساب المتوسط

$$avg = (f(xi,yi) + f(xi+1,y`i+1))/2$$

نحسب من جديد **yi+1 (Corrector)**

$$yi+1 = yi + w* avg$$

3-5 طريقة رنج كوتا Runge-Kutta Method

تتميز بدقتها وجودتها

$$Yi+1= yi + (k1+2k2+2k3+k4)/6$$

حيث

$$K1 = w* f(xi,yi)$$
$$K2= w* f(xi+w/2,yi+k1/2)$$
$$K3 = w* f(xi+w/2,yi+k2/2)$$
$$K4 = w* f(xi+w,yi+k3)$$

وبأخذ السؤال الذي يقول **أوجد حل مسألة القيم الذاتية** :

$dy/dx = -xy$; $x0 = 0$, $y0 = 1$

وذلك عند $x` = 1$ **علما بأن** $w=0.1$ **(قارن بقيم** $(e^{(-x^2/2)})$

سنقوم بحله بكل الطرق المذكورة آنفاً

حيث $f(x,y)=-xy$

أ ويـلـر

$$Yi+1 = yi - w * xi*yi$$

بتفاضل الدالة **f** وزيادة حد للسابقة نحصل على <u>**امتداد اويلر**</u>

$$Y`=-xy \text{ à } y``=-xy`+t(-1) = -xy`-y$$

بالتعويض عن قيمة **y`**

$$Y`` = -x(-xy)-y = x^2y-y= y(x^2-1)$$

وبذلك نحصل على امتداد اويلر

$$Yi+1 = yi - w * xi*yi + w^2*yi*(xi^2-1)/2$$

بتفاضل **f** مرة أخرى وزيادة حد نحصل على <u>**اويلر الاكثر امتداد**</u>

$$y``= y(x^2-1) \text{ à } y```=y(2x)+(x^2-1)y`$$

بالتعويض عن قيمة **y`**

$$Y`` = 2xy+(x^2-1)(-xy) = xy(3-x^2)$$

وبذلك نحصل على اويلر الاكثر امتدادا

$$Yi+1 = yi - w * xi*yi + w^2*yi*(xi^2-1)/2 + w3*yi*xi*(3-xi^2)/6$$

أما **اويلر المعدلة** فالكتالي:

Y`I+1=yi –w*xi*yi
Avg= (-xi*yi – xi+1*y`i+1)/2
Yi+1 = yi + w* avg

برنج كوتا :

K1= w*(-xi*yi)
K2= w* (-(xi+w/2)*(yi+k1/2))
K3 = w* (-(xi+w/2)*(yi+k2/2))
K3=w*(-(xi+w)*(yi+k3))
Yi+1= yi + (k1+2k2+2k3+k4)/6

وفي كل طريقة نكرر العمل عدد **n** من المرات بعدد الفترات المأخوذة .

المخطط والخوارزمية التاليين لطريقة رنج كوتا

قائمة بالمتغيرات المستخدمة في الخوارزمية والبرنامج

I: متغير للتكرار

N: عدد التكرار أي عدد الفترات

Xi,yi : قيم **x** و **y** المعطاة (الابتدائية)

xnew: قيمة **x** التالية إي في نهاية الفترة الجزئية

Ynew: قيمة **y** عند **x** التالية

K1,k2,k3,k4: معاملات رنج كوتا

W: عرض الفترة الواحدة

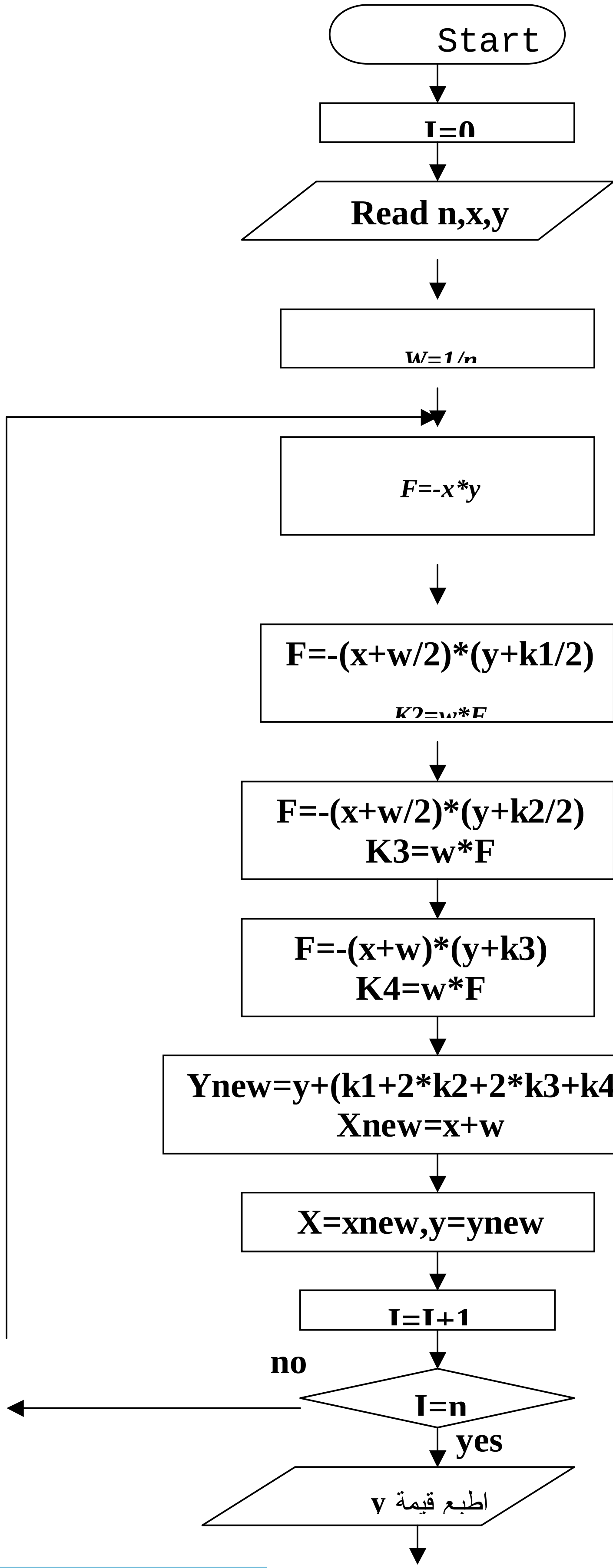
F: قيمة الدالة (التفاضل) عند **x,y**

الخوارزمية:

14. أبدا
15. اجعل **I=0**
16. ادخل عدد الفترات **n** وقيمة **x,y** الابتدائيات
17. احسب **w=1/n**
18. احسب **F=-x*y**
19. احسب **k1=w*F**
20. احسب **F=-(x+w/2)*(y+k1/2)**
21. احسب **k2=w*F**
22. احسب **F=-(x+w/2)*(y+k2/2)**
23. احسب **k3=w*F**
24. احسب **F=-(x+w)*(y+k3)**
25. احسب **k4=w*F**
26. احسب **ynew=y+(k1+2*k2+2*k3+k4)/6**
27. احسب **xnew=x+w**
28. اجعل **x=xnew** و **y=ynew**
29. اضف 1 إلى **i** **(I=I+1)**
30. هل **I=n** إذا كان لا ارجع إلى 5
31. اطبع قيمة **ynew**
32. النهاية

المخطط الانسيابي

Flow Chart



End

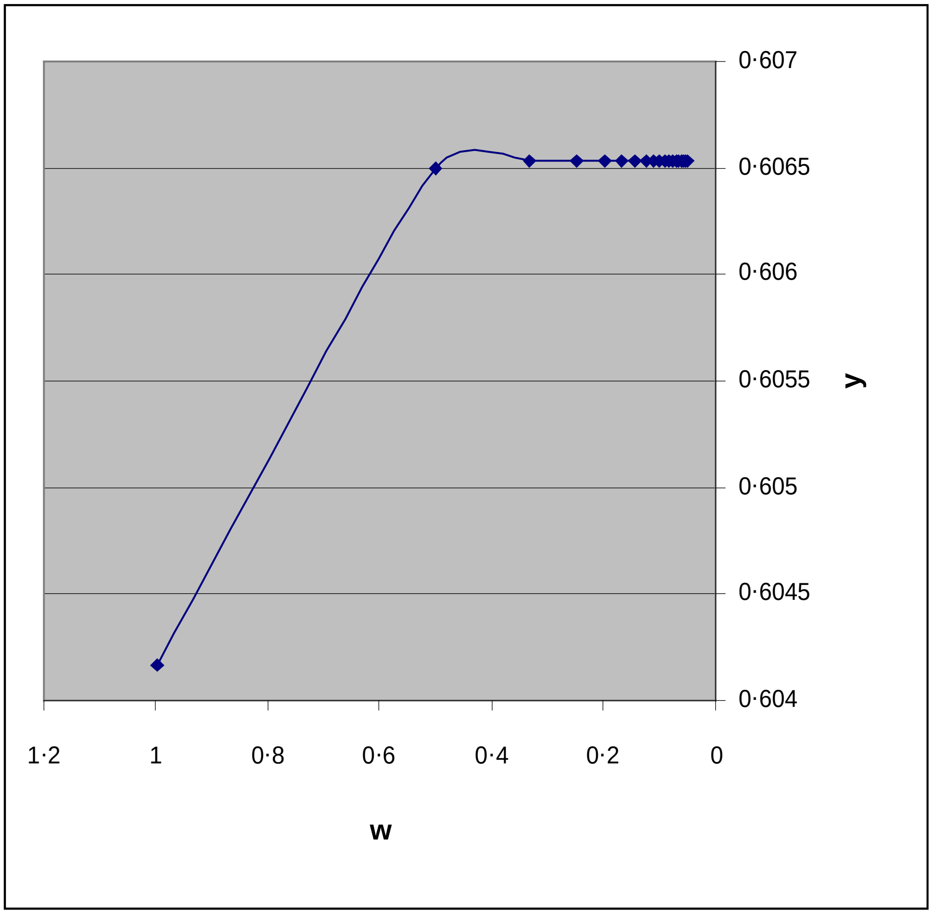
(شكل 3-1)

ملاحظة :

- البـــرنامج الاول هـــو البـــرنامج الرئيسي الذي يستخدم طريقة رنج كوتا بالعرض المطلوب في السؤال **w=0.1**
- البرنامج الثاني جعلت فيه **w** متغيرة بحيث تم أخذ فترات مختلفة
- البرنامج الثالث استخدمت طريقة اويلر وإمتداداتها
- البرنامج الرابع طريقة اويلر المعدلة
- من الممكن طبعا ادماج الطرق جميعا في برنامج واحد

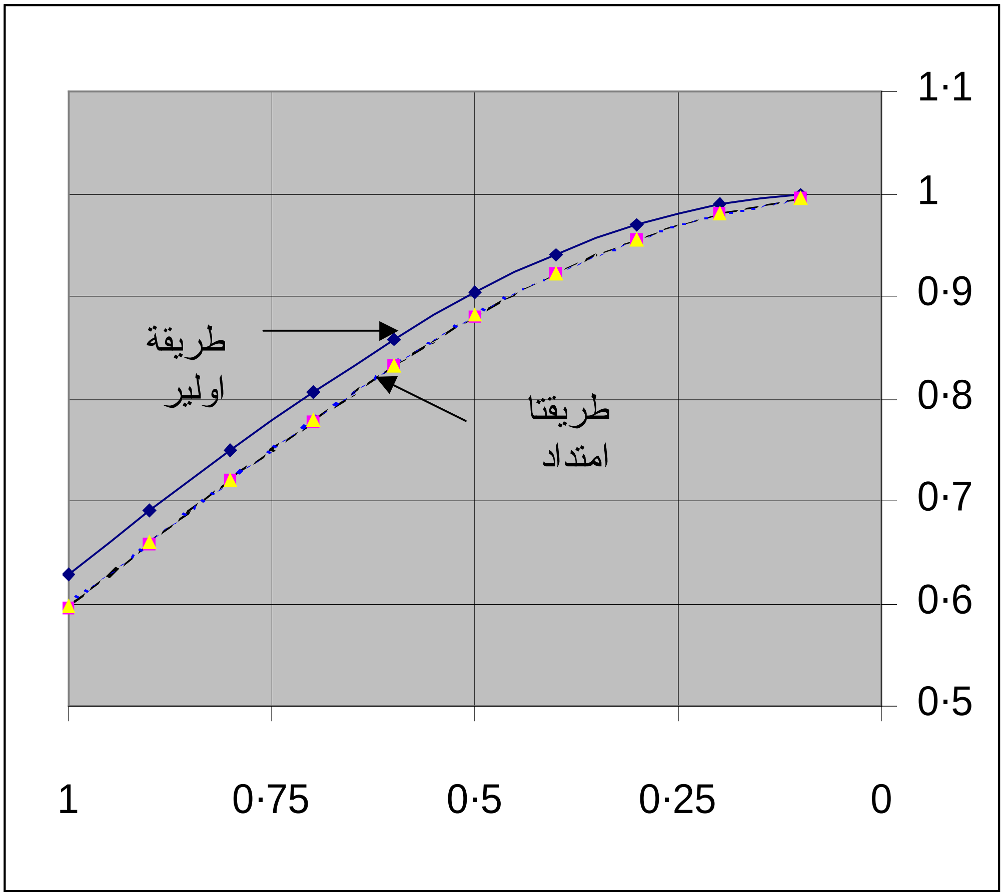
التعليـــــــق

- طريقة رنج كوتا هي أدق الطرق
- اويلر المعدلة اعطت نتائج ادق من اويلر الاخريات
- أمـــا اويلر وامتداداتها فان الاكثر امتدادا كانت أدق ثم امتداد اويلر وأخيرا اويلر
- بزيادة عدد التكرار أي بتصغير **w** تكون النتائج أدق



(شكل 3-2)

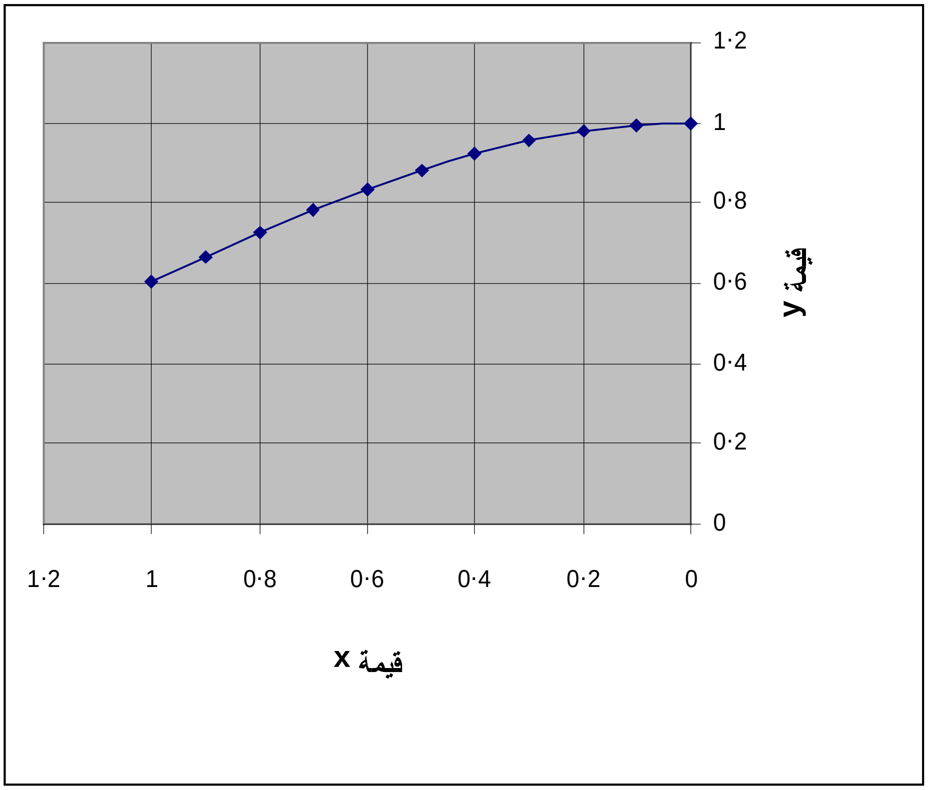
العلاقة بين مقدار **w** وقيمة **y** الناتجة حيث يتضح التقارب نحو الحل الصحيح بتقليل الفترات



(شكل 3-3)

المحور السيني يمثل قيمة **x** من 0 إلى 1 بزيادة قيمة **w** والمحور الصادي هو قيمة **y**

هذا الشكل يبين الطرق لثلاثة.



(شكل 3-4)

منحنى الدالة الأصلي مع القيم الناتجة من طريقة رنج كوتا واويلر المعدلة
حيث نلاحظ التطابق بينهم

3-6 طريقة الرمي

مقدمة

تستخدم طريقة الرمي لإيجاد حل المعادلات التفاضلية من الرتبة الثانية , ولذلك يجب علينا معرفة ثابتين وعادة ما تكون الثوابت إما الدالة ومشتقتها الأولى عند نقطة البداية (مسألة القيمة الابتدائية) أو الدالة ومشتقتها عند نقطتين مختلفتين (مسألة القيم الحدية) كمسألتنا هذه.

وتتلخص طريقة الرمي في:

- نفترض الشرط الناقص من شروط مسائل القيم الابتدائية وهو هنا تفاضل الدالة عند **x=1**
- اوجد الحل بالافتراضات الجديدة **(x`=3)** وقارنه مع الشرط المعطى عند تلك النقطة.
- أعد تغيير القيم الابتدائية أي الفرض الذي فرضته حتى نحصل على المطلوب وهو (في هذه المسألة **y(3)=10.0179**)

وبأخذ السؤال الذي يقول **أوجد حل مسألة القيم الذاتية** :

$D^2y/dx^2 = y \;;\quad y(1) = 1.1752 \;,\quad y(3) = 10.0179$

أولا: نأخذ قيمة افتراضية لـ**y`** عند **x=1** ثم نستخدم طريقة رنج كوتا لإيجاد قيمة **y** عند **x=3** ز

ثانيا: نفترض قيمة أخرى لـ**y`** عند **x=1** ونعيد استخدام طريقة رنج كوتا لنحصل على **y** عند **x=3** .

ثالثا نستخدم التوليد الخطي لإيجاد قيمة **y`** أخرى كالتالي:

Y`	Y	∇Y
R1	g1	g2-g1
R2	g2	
D	xo[2]	

$Y=y0+p\nabla y0$

$P=(xp-x0)/h$

باستخدام متغيراتنا

$H=r2-r1$

$\nabla Y=g2-g1$

$P=(d-r1)/(r2-r1)$

$Y=X0[2]=g1+(d-r1)/(r2-r1)*(g2-g1)$

ثم نستخدم رنج كوتا من جديد لإيجاد **y**عند **x=3** ونقارن بقيمة **y** الحقيقية المعطاة

عند**x=3** حتى نحصل على الحل الصحيح.

عدد التكرارات 20 تكرار وهو ناتج من n=(x-x0)/w أي **n=(3-1)/0.1=20**

قائمة بالمتغيرات المستخدمة في الخوارزمية والبرنامج

I,iter: متغير للتكرار

N: عدد التكرار أي عدد الفترات

tstart : قيمة **x** (الابتدائية)

xstart: قيمة **y** الابتدائية

to: متغير يأخذ قيمة **x** ويتزايد بمقدار **w**

Xwrk: مصفوفة بها معاملات رنج كوتا **K1,k2,k3,k4**

h: عرض الفترة الواحدة **w**

F: قيمة الدالة (التفاضل)

Tol: مقدار الخطأ المسموح به

G1,g2 : فيم **y`** الافتراضيات

X[1]: هذا العنصر من المصفوفة يحوي قيمة **y**

X[2]: وهذا يحوي قيمة **y`**

R1,r2 : قيمة **y** عند **x=3** في حالات **y`** المختلفة

Rksyst(): برنامج فرعي لحل المعادلة بطريقة رنج كوتا

Derives(): برنامج فرعي لحساب التفاضلات

الباب الرابع

الحل العددي للمعدلات لنظام المعدلات الخطية

(أكثر من مجهول واحد)

كنا قد تناولنا في المقالات السابقة، طرق حل معادلة واحدة ذات مجهول واحد فقط،، سواء أن كان من المرتبة الأولى أو الثانية فما فوق، و طرحت أهم الطرق المستخدمة في الحل، و في هذا الدرس نطرح طريقة حل أكثر من معادلة تحتوي على أكثر من مجهول، و لهذه الطريقة أهمية بالغة في التحليلات المختلفة, و تتميز بكثرة الاستعمال, و بالأخص في تطبيقات الحاسوب.

4-1 طريقة جاكوبي لحل مسائل القيم الذاتية

مسائل القيم الذاتية تتلخص في ايجاد قيم λ وهي القيم الذاتية و x وهي المتجهات الذاتية ونستخدم لذلك طريقة جاكوبي.

$$Ax= \lambda I x$$

$$\begin{vmatrix} a11 & a12 & a13 \\ a21 & a22 & a23 \\ a31 & a32 & a33 \end{vmatrix} * \begin{vmatrix} x1 \\ x2 \\ x3 \end{vmatrix} = \begin{vmatrix} \lambda 1 & 0 & 0 \\ 0 & \lambda 2 & 0 \\ 0 & 0 & \lambda 3 \end{vmatrix} * \begin{vmatrix} x1 \\ x2 \\ x3 \end{vmatrix}$$

حيث A مصفوفة مربعة متجانسة

طريقة جاكوبي :

بهذه الطريقة نريد الوصول إلى المعادل

$$Bx= \lambda X$$

حيث λ و B مصفوفتان قطريتان

$$B= \begin{vmatrix} b11 & 0 & 0 \\ 0 & b22 & 0 \\ 0 & 0 & b33 \end{vmatrix}$$

سنستخدم مصفوفة ذات ثلاثة أبعاد كعينة

وطريقة جاكوبي تقوم بدوران المصفوفة بزاوية θ في مستوى آخر

$$X=Tx$$

حيث T مصفوفة الدوران ولأنها في ثلاثة ابعاد توجد منها ثلاثة أنواع

$$T1= \begin{vmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{vmatrix}$$

$$T2= \begin{vmatrix} \cos\theta & & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{vmatrix}$$

$$T3= \begin{vmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{vmatrix}$$

بما أن

$Ax=\lambda x$

$ATx=\lambda Tx$

بما أن ضرب المصفوفة في معكوسها =1 لذلك نضرب الطرفين في معكوس T

$T^{-1}ATx=\lambda T^{-1}Tx=\lambda x$

ولكي نجعل $T^{-1}AT$ مصفوفة فطرية B نختار الزاوية المناسبة أي أن يكون الحد الغير الطري في المفوفة الناتجة من الضرب يساوي الصفر

$$a12(\cos 2\theta-\sin 2\theta)+\cos\theta\sin\theta(a22-a11)$$

أي

$$\tan(2\theta)=2*a12/(a11-a22)$$

$$\theta=\tan-1(2*a12/(a11-a22))/2......(1)$$

وللحصول على المتجهات الذاتية نقوم بضرب مصفوفات الدورات المستخدمة من البداية وحتى الحصول على القيم الذاتية.

$$V=T1*T2*...Tn$$

حيث n عدد الدورانات.

في المثال المعني

$$A=\begin{vmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{vmatrix}$$

الحل

1. نأخذ أكبر عنصر غير قطري في المصفوفة وهو هنا 1-
2. مما سبق نأخذ إما T1 أو T3 لأن أكبر عنصر مكرر في مكانين
3. نحسب فيمة الزاوية من المعادلة (1) وتساوي
   $\theta=\tan-1(-2/(2-2)=-\Pi/2$
4. نعين المصفوفة الناتجة من الضرب $T^{-1}AT$ وتصبح هي A
5. نضرب مصفوفة الوحدة في T المرة الأولى ثم الناتج في T المستخدمة أخيرا. وذلك للحصول على المتجهات الذاتية.
6. نعيد الخطوات 1و2و3 على المصفوفة الجديدة حتى نصل إلى أن تكون قطرية .

الخوارزمية:

33. البداية
34. اقرأ المصفوفة المتجانسة **A** ذات البعدين 3×3
35. أوجد أكبر عنصر غير قطري في المصفوفة
36. وفق موقع العنصر الاكبر اختر أي **T** نستخدم
37. أوجد قيمة الزاوية θ من المعادلة (1)
38. أوجد معكوس **T**
39. أوجد حاصل ضرب معكوس **T** في **A** في **T** واجعل الناتج في **A**
40. هل العناصر غير القطرية < نسبة خطأ معينة (مثلا 0.0001)

... إذا

لا : انتقل إلى 3

نعم: اطبع العناصر القطرية **a11,a22,a33**

41. النهاية

ملاحظة:

الخوارزمية والمخطط الانسيابي مختصران حيث أن عملية قراءة المصفوفة لها مراحل معروفة يمكن رؤيتها في البرنامج وأهملتها هنا لعدم الإطالة.

كذلك عملية ضرب المصفوفات تركت لعدم الإطالة ولكنها واضحة في البرنامج.

اهم متغيرات ودوال البرنامج:

Biger(): تحدد أكبر عنصر غير قطري.

Equal(): تساوي المصفوفة الأولى بالثانية.

Tandinvers(): تحدد أي **T** نستخدم مع معكوسها . **Write_mat**:طباعة المصفوفة

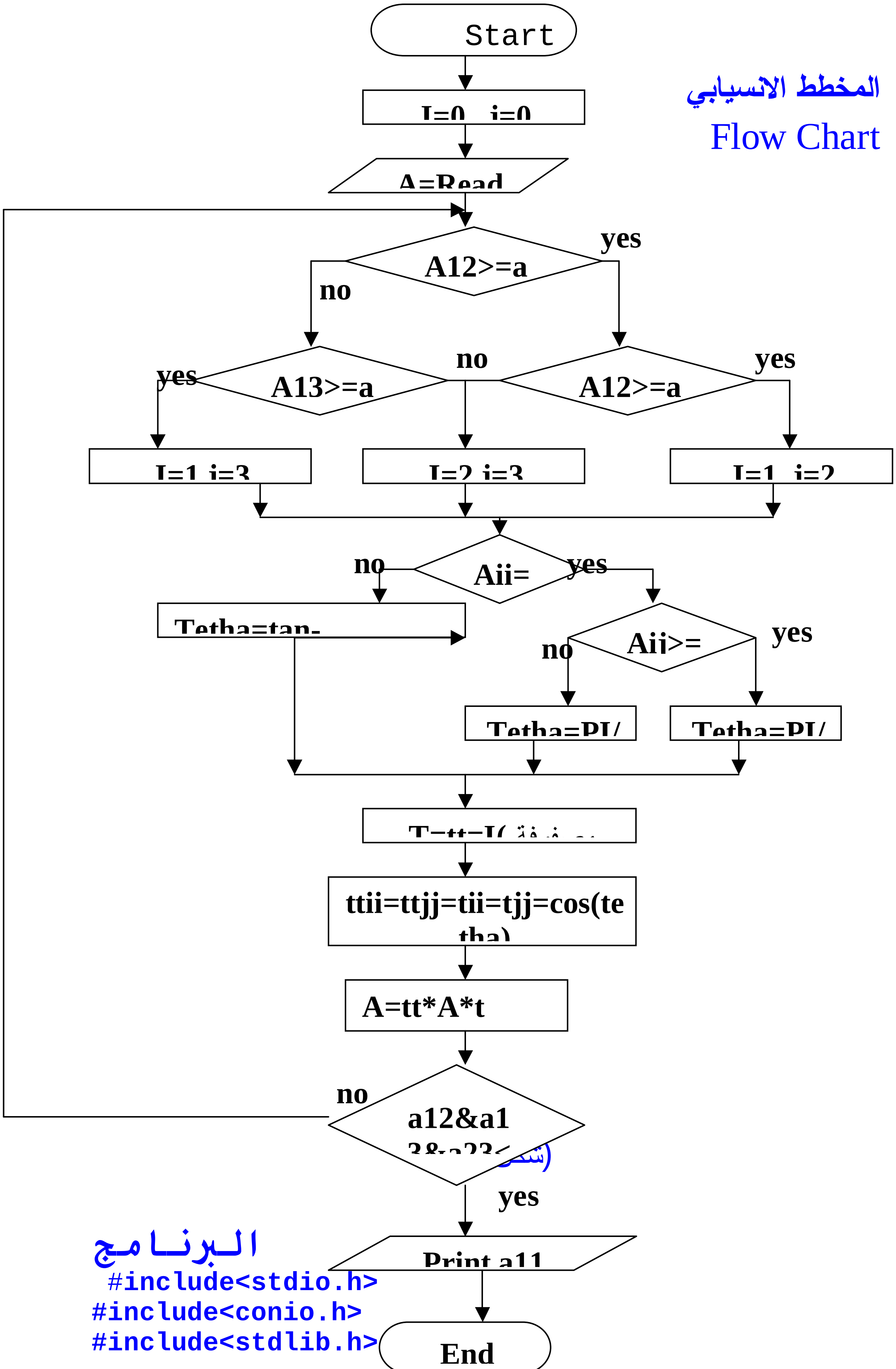
Mull_mat: تضرب مصفوفتين . **read_mat**: تقرأ المصفوفة.

V: مصفوفة المتجهات. **A**: المصفوفة الرئيسية. **T**: مصفوفة الدوران.

K: التي في البرنامج الرئيسي تعني عدد الدورانات.

المخطط الانسيابي

Flow Chart



البرنامج

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
```

```
#include<math.h>
#include<iostream.h>
#define max 3
int i,j,k,n;
void equal(float s[][3],float c[][3]);
void input_mat(float a[max][max]);
void mull_mat(float aa[][max],float
b[][max],float c[][max]);
void write_mat(float a[][max]);
void biger(float a[][max]);
void tandinvers(float t[][3],float tt[][3],float
a[][max]);
FILE *in,*out;
main()
{ int k;
float
d[max][max],a[max][max],c[max][max],tt[3][3],t[3]
[3],v[max][max]={1,0,0,0,1,0,0,0,1};
clrscr();
n=3;  k=0;
in=fopen("jacread.fil","r");
input_mat(a);
fclose(in);
out=fopen("jacwrite.fil","w+");
fprintf(out,"The Array of A =\n");
write_mat(a);
fprintf(out,"\n");
while( (fabs(a[0][1])>0.0001) ||
(fabs(a[0][2])>0.0001) || (fabs(a[1][2])>0.0001)
)
{ k=k+1;
  biger(a  ;(     //      find biger element of A
  fprintf(out,"\nCycle no. %d & The biger element
: %5.4f\n",k,a[i][j]);
  tandinvers(t,tt,a ;(     //      find Angle , T
& invers T
  fprintf(out,"T = \n");
  write_mat(t);
  mull_mat(tt,a,d);   // multiplicat invers T in
A
  mull_mat(d,t,a);    // multiplicat  A in T
  fprintf(out,"A = \n");
  write_mat(a);
  mull_mat(v,t,c);     //clculat Eigenvectors
```

```
  equal(v,c);
}
fprintf(out,"The lamda is ");
for(i=0;i<3;i++){
fprintf(out,"\nJ%d=%5.4f\n V= ",i,a[i][i]);
for(j=0;j<3;j++)
fprintf(out,"%5.4f  ",v[j][i]);
}
fclose(out);
return(0)   ;
}
// function to find angle , T and invers T
void tandinvers(float t[][3],float tt[][3],float
a[][max])
{ float x,z,y,tunit[3][3]={1,0,0,0,1,0,0,0,1};
  if(a[i][i]==a[j][j]) if(a[i][j]>=0) x=3.1416/4;
             else  x=-3.1416/4;
  else x=(atan(2*a[i][j]/(a[i][i]-a[j][j])))/2;
  y=cos(x); z=sin(x);
  equal(t,tunit);   // t= {1,0,0,0,1,0,0,0,1}
  equal(tt,tunit);
  // select T1 , T2 or T3 consider biger element
Y
for(i=0;i<n;i++)
for(j=0;j<n;j++)
fscanf(in,"%f",&a[i][j]);
}
void biger(float a[][max])
{
   if(fabs(a[0][1])>=fabs(a[0][2(([
    if(fabs(a[0][1])>=fabs(a[1][2]))  {i=0;j=1;}
    else                {i=1;j=2;}
   else if(fabs(a[0][2])>=fabs(a[1][2]))
{i=0;j=2;}
    else        {i=1;j=2;}
}
void mull_mat(float aa[][max],float
b[][max],float c[][max])
{int k;
 for(i=0;i<n;i++)
  for(j=0;j<n;j++)
   { c[i][j]=0;
     for(k=0;k<n;k++)
       c[i][j]=c[i][j]+aa[i][k]*b[k][j]    ;
```

```
        }
    }
    void write_mat(float c[][max])
    {
      for(i=0;i<n;i++)
      {
        for(j=0;j<n;j++)
          fprintf(out,"%5.4f\t",c[i][j]);
        fprintf(out,"\n");
}}
```

الإدخالات و **النتائج**

```
The Array of A=
2.0000   1.0000-  0.0000
1.0000-  2.0000   1.0000-
```

2.0000 1.0000- 0.0000

```
Cycle no. 1 & The biger element : -1.0000
T =
0.7071   0.7071   0.0000
0.7071-  0.7071   0.0000
0.0000   0.0000   1.0000
A =
3.0000   0.0000   0.7071
0.0000   1.0000   0.7071-
0.7071   0.7071-  2.0000

Cycle no. 2 & The biger element : 0.7071
T =
0.8881   0.0000   0.4597-
0.0000   1.0000   0.0000
0.4597   0.0000   0.8881
A =
3.3660   0.3251-  0.0000-
00.3251- 1.0000   0.6280-
0.0000-  0.6280-  1.6340

Cycle no. 3 & The biger element : -0.6280
T =
1.0000   0.0000   0.0000
0.0000   0.8517   0.5241-
0.0000   0.5241   0.8517
A =
```

```
3.3660   0.2768-  0.1704
0.2768-  0.6136   0.0000
0.1704   0.0000   2.0204

Cycle no. 4 & The biger element : -0.2768
T =
0.9951   0.0991   0.0000
0.0991-  0.9951   0.0000
0.0000   0.0000   1.0000
A =
3.3936   0.0000-  0.1695
0.0000   0.5860   0.0169
0.1695   0.0169   2.0204

Cycle no. 5 & The biger element : 0.1695
T =
0.9927   0.0000   0.1207-
0.0000   1.0000   0.0000
0.1207   0.0000   0.9927
A =
3.4142   0.0020   0.0000
0.0020   0.5860   0.0168
0.0000   0.0168   1.9998

Cycle no. 6 & The biger element : 0.0168
T =
1.0000   0.0000   0.0000
0.0000   0.9999   0.0119
0.0000   0.0119-  0.9999
A =
3.4142   0.0020   0.0000
0.0020   0.5858   0.0000-
0.0000   0.0000-  2.0000

Cycle no. 7 & The biger element : 0.0020
T =
1.0000   0.0007-  0.0000
0.0007   1.0000   0.0000
0.0000   0.0000   1.0000
A =
3.4142   0.0000   0.0000
0.0000   0.5858   0.0000-
0.0000   0.0000-  2.0000
The lamda is
```

```
J0=3.4142
 V= 0.5000  -0.7071  0.5000
J1=0.5858
 V= 0.5000  0.7071  0.5000
J2=2.0000
 V= -0.7071  -0.0000  0.7071
```

الاستنتاج والتعليق

- تم استخدام عدة برامج فرعية لقراءة وطباعة المصفوفة ولحساب الضرب وغيره
- يختلف عدد الدورانات باختلاف الدقة المختارة للخطأ المسموح به فكلما زادت الدقة كلما زاد عدد الدوران فالعلاقة طردية (عند 0.0001 عدد 7 دورانات)
- نلاحظ في حالة تساوي قيمتين كأكبر قيمة فإن الاختلاف في أخذ T يؤدي إلى نفس الحل ولكن ليس بنفس الترتيب .
- استخدمت شرطا خاصا لإيجاد قيمة الزاوية عندما يكون المقام بصفر
- لاحظ أن من موقع أكبر عنصر نستطيع تحديد T فلو كان موقع أكبر عنصر I=l,j=k فإننا نضع all=cos, akk=cos , alk=-sin , akl=sin
- استخدمت مصفوفة الوحدة لوضعها في مصفوفة الدوران قبل تحميلها كما سبق.

4-2 طريقة لا جرانج للتوليد Lagrange's Interpolating Polynomial

تعتبـــر طـــريقة لاجــرانج طريقة جيدة لتوليد قيم جديدة لكثيرات الحـــدود وكـــذلك لايجاد صيغة كثيرة الحدود . وهي تصلح لأي عدد من النقاط. وتعتمد درجة كثيرة الحدود الناتجة على عدد النقاط المستخدمة.

درجة كثيرة الحدود = **n-1** حيث **n** عدد النقاط

فلـــو كانت عدد النقاط 3 فستكون كثيرة الحدود من الدرجة الثانية وهكذا وتكـــون عدد معاملات لاجرانج **L** تعتمد على عدد النقاط وتكون قيمة **L** على الصورة:

$$\mathbf{Li}=\frac{(x-x_1)(x-x_2)......(x-x_{i-1})(x-x_{i+1}).......(x-x_n)}{(x_i-x_1)(x_i-x_2)......(x_i-x_{i-1})(x_i-x_{i+1}).......(x_i-x_n)} \quad ....\mathbf{I}$$

=<u>حاصل ضرب حاصل طرح قيمة **x**من كل النقاط عدا النقطة الحالية</u>

حاصل ضرب حاصل طرح النقطة الحالية من كل النقاط عدا نفسها

ثم لإيجاد المعادلة النهائية :

$$\mathbf{Y=L(x)} = \sum_{i=1}^{n} Li(x)*yi$$

ومن الملاحظ أنه لا يمكن بسهولة أو قد يتعذر حلها يدويا خاصة عندما تكثر عدد النقاط ويزداد صعوبة عندما تكون قيم النقاط بالأرقام العشرية. لذلك تحل باستخدام الحاسوب. وسنبرمج طريقة لاجرانج باستخدام لغة **C**

.

قائمة بالمتغيرات المستخدمة في البرنامج

I,j : متغيرات للتكرار

N : عدد النقاط المدخلة

Xp : النقطة المراد إيجاد قيمة **y** عندها

Xi,yi : النقاط المدخلة

C: حاصل ضرب طرح قيمة **xp** من النقاط أي البسط في القانون **(I)**

D: حاصل ضرب طرح قيمة **xi** الحالية من جميع النقاط أي مقام القانون **(I)**

L: معامل لاجرانج

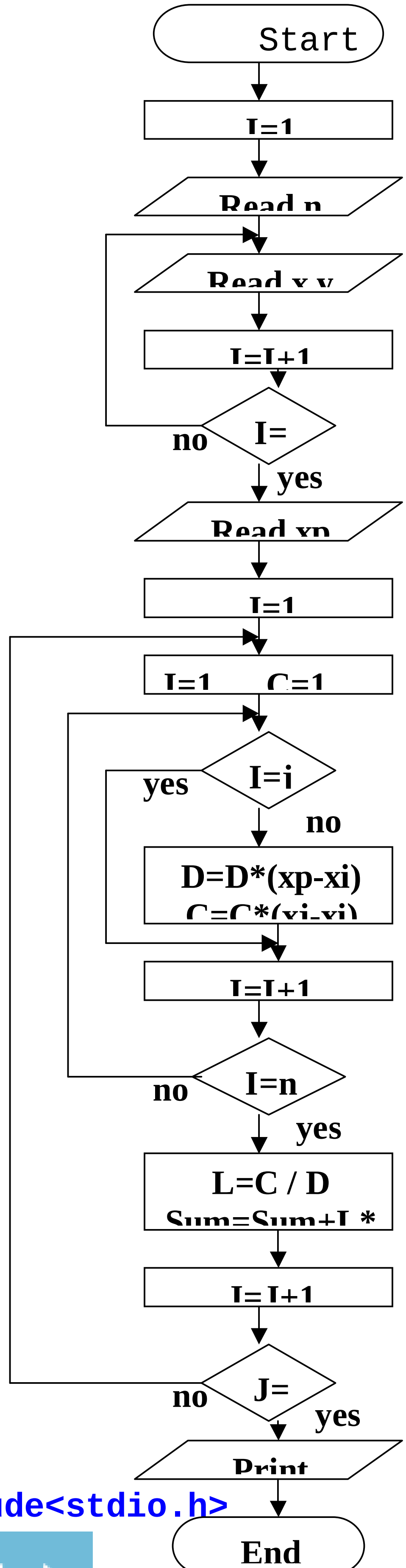
Sum: مجموع حاصل ضرب لاجرانج في قيمة **y**

الخوارزمية:

42. ابدا
43. ادخل قيمة **n** (عدد النقاط)
44. ادخل النقاط المتكونة من قيم **x,y**
45. ادخل قيمة **xp** المراد إيجاد **y** عندها
46. اجعل قيمة **C** و **D** تساوي 1
47. هل ترتيب **L** الحالي يساوي ترتيب النقطة الحالية **(x,y)**
... إذا كان نعم انتقل لــ 8
48. احسب **C=C*(xp-xi)** و **D=D*(xj-xi)**
49. انتقل للعنصر التالي ويمثلها **i**
50. هل انتهت النقاط **(I=n)**... إذا كان لا... انتقل لــ 7
51. احسب **L** كالتالي **L=C/D**
52. اضرب **L** في **Y** واجمعها على **sum** (للنقطة الحالية)
53. انتقل للمعامل التالي ويمثلها **j**
54. هل انتهت النقاط **(j=n)**... إذا كان لا... انتقل لــ 5
55. اطبع **sum**
56. النهاية

المخطط الانسيابي

Flow Chart



البرنامج بلغة c

```
# include<stdio.h>
```

```
# include<conio.h>
void main()
{int i,j,n;
 float L[9],sum,xp,x[9],y[9],C,D;
 sum=0;
 printf("Enter n value\n ");
 scanf("%d",&n);
 printf("\n Enter x and y \n ");
 for(i=0;i<n;i++)
   scanf("%f %f",&x[i],&y[i]);
 printf("\n Enter point of x   ");
 scanf("%f",&xp);
 clrscr();
 printf("Lagrange method\n\n");
 printf("\n x ");
 for(i=0;i<n;i++)  printf("%4.2f\t",x[i]);
 printf("\n y ");
 for(i=0;i<n;i++)  printf("%4.2f\t",y[i]);
printf(" xp= %4.2f\n",xp);
printf\_________________________________")
n");
 for(i=0;i<n;i++)
   { C=1; D=1;
   for(j=0;j<n;j++)
      { if(i!=j) {
      C=C*(xp-x[j]) ;
       D=D*(x[i]-x[j]);
      }      }
     L[i]=C/D;
     sum=sum+L[i]*y[i];
     printf("\n L(%d)= %f ",i,L[i]);
    }
 printf("\n\______________________________
n");
 printf("\nL=L1*y1+l2*y2+....\n");
 printf\______________________________")
n\n");
 printf("L= %f ",sum);
```

```
}
```

الإدخالات و النتائج

مثال 1

Lagrange method

```
 x  0.00    1.00    2.00    3.00    4.00
 y  0.00    2.00    8.00    18.00   32.00     xp=
5.00
```

```
 L(0)= 1.000000
 L(1)= -5.000000
 L(2)= 10.000000
 L(3)= -10.000000
 L(4)= 5.000000
```

```
L=L1*y1+l2*y2+....
```

```
L= 50.000000
```

مثال 2 a بإعطاء بيانات أخرى تعبر عن sin(x) تكون النتائج

....

Lagrange method

```
 x    0.00    30.00   60.00           90.00
 y    0.00    0.50    0.8660254038    1.00      xp=
40.00
```

```
 L(0)= -0.061728
 L(1)= 0.740741
 L(2)= 0.370370
 L(3)= -0.049383
```

L=L1*y1+l2*y2+....

L= 0.641738

مثال 2 **b** بإعادة الإدخالات ولكن مبعثرة

....

Lagrange method

x 90.00 30.00 0.00 60.00
y 1.00 0.50 0.00 0.8660254038 **xp=**
40.00

L(0)= -0.049383
L(1)= 0.740741
L(2)= -0.061728
L(3)= 0.370370

L=L1*y1+l2*y2+....

L= 0.641738

مثال 2 c بإعادتها بعدد نقاط أقل

....

Lagrange method

x 0.00 30.00 60.00
y 0.00 0.50 0.8660254038 **xp=**
40.00

L(0)= -0.111111
L(1)= 0.888889

```
 L(2)= 0.222222
```

```
L=L1*y1+l2*y2+....
```

```
L= 0.636895
```

مثال 2 d بنقاط أخرى ولكن بنفس العدد السابق

....

Lagrange method

```
x   30.00   60.00          90.00
y   0.50    0.8660254038   1.00      xp= 40.00
```

```
 L(0)= 0.555556
 L(1)= 0.555556
 L(2)= -0.111111
```

```
L=L1*y1+l2*y2+....
```

```
L= 0.647792
```

الاستنتاج

*عند إعطاء النقاط لا يهم ترتيبهم فالناتج واحد أي بعثرة النقاط لا يؤثر على هذه الطريقة.

*من المعروف أنه بزيادة عدد النقاط تزداد درجة الحدودية ولكن لا يؤثر عـــدد النقاط أو تزايدها في الدوال قليلة الحدود كما بالمثال الأول فالدالة $2x^2$ التـــي تعبـــر عنها النقاط المعطاة أعلاه مهما زدنا عدد النقاط عن الثلاثة فلا تأثير **ولكن** في كثيرات الحدود كما بالمثال الثاني **sin(x)** فإنه

بـــزيادة عـــدد الـــنقاط تـــزداد درجة الحدودية و نحصل على نتيجة أدق وبالتالي خطأ أقل .

*مجموع معاملات لاجرانج تساوي 1.

*نلاحـــظ فـــي المـــثال 2 **d-c** انه بنفس عدد النقاط ولكن بدل أن نأخذ **x0,x1,x2** أخذنا **x1,x2,x3** فاختلف بذلك الناتج عن اخذ أربع نقاط 2-**a** . ففـــي 2- **c** كانـــت **xp** قريبة من النقطة الثالثة فنقصت **L** وعندما كانـــت **xp** قريبة من النقطة الأولى في 2-**d** زادت **L** فهي تتقارب إلى تكتل النقاط ويتضح ذلك بالرسم.

بالبرنامج بلغة الفورتران

```
Dimension   x(50),f950)
        Write (*,*)'enter value of polynomial'
         Read (*,*) n
         Write (*,*)'enter point of x that you find solution'
Read(*,*)x0
   Do 10,i=1,n+1
          Write (*,*)'x ( ',i,')','      ,'f( ',i,' ) = '
          Read(*,*)x(i),f(i)
  10  Continue
        Xp=0
Do  20,i=1,n+1
       t=1
             Do 30,j=1.n+1
If(i.ne.j)then
                     T=(t*(x0-x(j)))/(x(i)-x(j))
                   Else
                Endif
30    Continue
        Px=px+t*(f(i)
20     Continue
        Write(*,*)xp
         Stop
          End
```

3-4 حذف جاوس: طريقة التعويض الخلفي

قبل الدخول في التطبيق البرمجي ، أرى من الأهمية بمكان طرح المفهوم النظري لطريقة جاوس أو طريقة (التعويض الخلفي) .

الأساسيات النظرية

تقوم طريقة جاوس على تحويل المعدلات إلى نظام المصفوفات و التعامل معها على هذا الأساس كما الآتي :

A1x1 + a2x2 + a3x3 + …….anxn = c1

B1x1 + b2x2 + b3x3 + ... bnxn= c2

D1x1 +d2x2 +d3x3+ ……..dnxn=cn

نلاحظ ان لدينا عدد كبير من المعادلات يصل إلى ما قيمته n و عدد المجاهيل x1 وx2… xn هي المراد إيجاد قيمتها ، بينما القيم a1 a2 an , b1 b2 bn وكذلك d1 d2 d3 ….dn

المرافقات للمجاهل، فإذا رتبنا هذه في المصفوفة التالية:

a1,1 a1,2 ….. a1n

b2,1 b2,2 …… b2n

dn1 dn2 .. …. dnn

فهذه تسمى مصفوفة المرافق و تبدأ بالقيمة المرافقة للمتغيرات و تتكون من صفوف و أعمدة كما هو معروف, بينما المصفوفة التالية:

X1

X2

.

.

xn

فهذه تسمى مصفوفة المجاهيل, و المراد إيجاد قيمها ؟، بينما المصفوفة التالية :

C1

C2

.

.

cn

تسمى مصفوفة القيم المطلقة، و بتعبير أخر حولنا المعادلات غلى ثلاث مصفوفات كي يمكن التعامل معها , الخطوة التالية هي إيجاد المصفوفة الموسعة (augmented matrix) عن طريق إدخال مصفوفة القيم المطلقة مع

مصفوفة المرافقات و الملاحظ عليها أن عدد الأعمدة أكثر من عد الصفوف بمقدار واحد , لتصبح كالآتي :

a1,1 a1,2 ..... a1n c1

b2,1 b2,2 ...... b2n c2

dn1 dn2 .. .... Dnn cn

(لاحظ أن الرقم a1,1 يعني العنصر الأول في الصف الأول من العمود الأول و الرقم a1,2 يعني العنصر الثاني من الصف الثاني في العمود الثاني و هكذا بقية العناصر)

الخطوة التالية هي جعل جميع العناصر التي تسبق عناصر القطر الرئيسي صفرا، و ذلك بإتباع الآتي:

F2,1=b2,1/a1,1

b2,1= b2,1 – f2,1*a1,1

b2,2 =b2,2 - f21*a1,2

b2,3=b3,2 – f1,2*a1,3

الذي قمنا به هو الآتي :

1. أوجدنا المعامل الذي يقوم بتحويل جميع العناصر التي تسبق عنصر القطر الرئيسي في الصف الثاني و ذلك بقسمة العنصر الذي رتبته (2,1) أي العنصر الأول في الصف الثاني على (1,1) أي العنصر الأول في الصف الأول
2. حولنا العنصر الذي رتبته (2,1) إلى صفر عن طريق الآتي : b21= b21 – f21*a11
3. طبقنا المعادلة السابقة على بقية عناصر الصف الثاني كي لا تتغير القيمة الفعلية للمصفوفة
4. نتبع بقية الخطوات بالنسبة لبقية الصفوف و نحول جميع العناصر التي تسبق القطر الرئيسي بنفس الطريقة

بعد أن اكتملت ملامح المصفوفة الموسعة نبدأ بالتعويض الخلفي و نبدأ بالصف الذي رتبته n (اىالصف الأخير) و تكون معادلته كالآتي:

0 0 …… dnn = x

و باعتبار أن قيمة آخر قيمة مجهولة قد علمت نجد القيمة التي تقل عنها بمقدار واحد هكذا نستمر حتى نصل إلى x1.

خوارزمية البرنامج

لكتابة خوارزمية البرنامج يلزم تعريف أربعة مصفوفات أولا

1. **البداية**
2. **عرف المصفوفات a(50,50) b (50, 50) f(50) x (50)**
3. **read n**
4. **do**

from I = 1 to n read elements of matrix a

5. **do**

m=n*(n-1)

6. **do**

from I=2 to m

from k=1 to i-1

if (a (k,k) ≠0 yes f(I,k)=a(I,k) / a(k,k)

7. **do**

from j = 1 to n +1

a(I,j)=a(I,j) – f(I,k)*a(k,j)

no

8. do

from l =I – 1 to I – 1

from j=1 to n+1

b(l ,j)= a(l , j)

a (l , j) = a (l + 1 , j)

a (l + 1 , j)= b (l , j)

9. do

from I = n to 1 step – 1

from j = I + 1 to n

s1= a(I, n +1)

x(i) = (s1 – s2)/ a(I , I)

s2 = 0

10. do

from I = 1 to n

write x(i)

11. stop

هذه هي خوارزمية البرنامج و يمكن تقسيمها إلى حلقات من اجل المزيد من الاستيعاب لها كالآتي :

1. حلقة قراءة قيم المصفوفة مضاف إليها عناصر مصفوفة القيم المطلقة أي ((n + 1

2. حلقة تكوين المصفوفة الموسعة و التي بها اختبار إذا كان العنصر المقابل للمعامل لا يساوي صفر فيتم ضرب المعامل في جميع عناصر المصفوفة ابتداء من الصف الثاني إلى الصف الأخير(الذي رتبته n)

3. حلقة قلب الأعمدة إلى صفوف في حالة أن العنصر المناظر للعنصر الذي سيقسم عليه يساوي صفر

4. حلقة التعويض الخلفي

البرنامج بلغة(الفورتران) FORTRAN language

```
Dimension a(50,50) ,b (50, 50 ) , f(50) , x(50)
Write(*,*)' enter  numbers of your elements (n)'
Read (*,*)n
Do 10,i=1,n
Do 20 , j=1 n+1
Write(*,*)'        ','a ( ',I,j,' )  = '
Read(*,*)a(I,j)
20      continue
10       continue
m=n*(n-1)/2
do 30,i=2,m
do 40,k=1,n-1
if(a(k,k).ne.0)then
do 40,j=1.n+1
a(i.j)=a(I,j)-f(I,k)*a(k,j)
40      continue
else
do 50,L=i-1,i-1
```

```
do 60,j=1,n+1
b(l,j)=a(l,j)
a(l,j)=a(l+1,j)
a(l+1,j)=b(l,j)
60 continue
50 continue
do 70,i=n,1,-1
do 80,j=i+1,n
s2=s2 + a(I,j)*(j)
80 continue
s1=a(I,n+1)
x(i)=(s1 – s2)/a(I,i)
s2=0.0
70 continue
do 90,i=1,n
write(*,*)' x  ( ',I, '  )  = ',x(i)
90 continue
stop
end
```

هذا الكود الذي يقوم بما سلف، و الآن لنبدأ في شرح الكود

شرح الكود

يبدأ البرنامج بقراءة قيم عدد المعدلات من خلال جملة n(*,*) read و التي تحدد عدد صفوف المصفوفة و عدد المجاهيل مع ملاحظة أن هذا النوع من الطرق يقوم بحل المعادلات التي تتساوى فيها عدد المعدلات مع عدد المجاهيل فقط أما عندما تكون عدد المعادلات اقل من عدد المجاهيل يلزم طرق أخرى و تعديلات مختلفة عن التي ورد ذكرها.

بعد تحديد قيمة المجاهيل يطلب البرنامج قراءة عناصر المصفوفة الموسعة بعدما تمت إضافة الحدود المطلقة إليها أي عدد n مضاف إليه واحد و يبدأ ذلك عن طريق الحلقة التي تتم الآتي

يجب قبل الدخول في الحلقة تحديد عدد العناصر التي سيتم حذفها من المصفوفة من اجل تكوين المصفوفة الموسعة و يتم ذلك عن طريق المعادلة التالية

$$M=n*(n-1)/2$$

و الحلقة الأولى تبدأ بالصفوف أي (i) ابتداء من الصف الثاني إلى قيمة m التي تحدد عن طريق المعالة السابقة, يلي ذلك لانتقال إلى الحلقة التالية و التي تحدد بالمعامل k التي يبدأ من القيمة 1 إلى I – 1 و المعامل k يحدد العنصر المناظر للعنصر الذي سيقسم عليه و هذا السبب الذي جعل الحلقة تحدد عند I – 1 أي الصف إلي يقل بمقدار واحد ليطابق العنصر تماما ، يلي ذلك اختبار هذا العنصر إذا كان لا يساوي صفر فيتم إيجاد المعامل الذي سيجعل العناصر التي تسبق عناصر القطر الرئيسي تساوي صفر، ليدخل البرنامج في الحلقة التي تليها و تقوم بإجراء التعديلات اللازمة على الصفوف من جعل العناصر التي تسبق القطر الرئيسي صفرا و إجراء نفس التغيير على بقية عناصر الصف عن طريق المعادلة التالية :

$$A(I,j) =a(I , j) –f (I, k)*a(k , j)$$

و جملة الاستمرارية continue لكي تتم العملية على كل عناصر المصفوفة كاملة

يتم هذا في حالة أن الإجابة للاختبار كانت بنعم أما إذا كانت الإجابة بلا و لأنه من غير المنطقي قسمة عدد على صفر لان الناتج سيكون كمية غير معرفة و لذا سيتم اتخاذ الإجراء التالي و هو قلب عناصر المصفوفة إلى أعمدة ، من المعروف هذا الإجراء لا يغير من قيمة المصفوفة شيء، و هو كالآتي :

$$b(l,j)= a(l , j)$$

$$a(l,j)=a(l+ 1 , j)$$

$a(l + 1, j)=b(l, j)$

أي كأننا قمنا بالاتي:

$C=a$

$A=b$

$C=b$

و هذا النوع من التبديلات ذو شهرة واسعة في تغيير القيم

الخطوة التالية هي الدخول في الحلقة التي تقوم بالتعويض الخلفي كالآتي:

ابتداء من الصف إلي رتبته n أي الأخير إلى الصف الأول بسالب خطوة واحدة إلى الخلف تم الحلقة التالية التي تبدأ من العمود الذي رتبته الصف الأخير منقوص منه واحد من اجل مراعاة أن العمود الأخير ليس من اصل المعادلة إنما أضيف من اجل تكوين المصفوفة الموسعة إلى العمود الأخير (n)

نكون متغير يحمل الاسم s2 الذي يقوم بقسمة العنصر الذي رتبته I , j على العنصر الذي رتبته j و كأننا قمنا بالاتي"

$2x=5 \quad x=5/2$

ووضعنا هذه القيمة في المتغير s2 و جملة الاستمرارية من اجل إتمام العملية على بقية العناصر، ننتقل إلى المتغير s1 الذي يقوم بحساب أو إيجاد العنصر الذي رتبته (I,n + 1) ، الخطوة الأخير في التعويض الخلفي هي إيجاد قيمة المجاهيل عن طريق المعادلة التالية:

$X(i)=(s1 - s2)/a(I , I)$

ثم تصفير المتغير s2 و إكمال الحلقة حتى إيجاد آخر مجهول و طباعة الناتج و الخروج من البرنامج

الباب الخامس

الملاءمة والانكفاء بواسطة البرمجة
FITTING METHOD

1-5 الملائمة والانكفاء FITTING METHOD

عــندما تعطـــى بيانات غير دقيقة نوعا ما أي تقريبية كالقياسات المعملية والهندسية فان هذه البيانات ترتبط بأخطاء مصدرها القياس أو الانسان أو .... . لذلك نلائم المنحنى الناتج من هذه البيانات حيث أن هذه البيانات تتبع معادلة معينة ولكــن قــد تلائم المنحنى لأكثر من شكل ولكن المنحنى المفترض أخذه يأتي من مــصدر البيانات نفسها هل تمثل المسئلة دالة تربيعية أو أسية أو ... وهكذا وبعد تحديــد الدالة يجب أن نجد أجود ملائمة **Best Fit** وهي تعني أن نجعل الأخطاء أقل مايمكن فنأخذ مجموع مربعات الأخطاء

$$S=\sum_{i=0}^{n} d^2$$

حيث

$$d_{\circ} = P_m(x)-yi$$

$$P_m(x) = a_0+a_1x+\dots\dots+a_mx^m = \sum a_j g_j(x)$$

وهذه الطريقة هي طريقة ملائمة المنحنيات باستخدام المربعات الصغرى

وتكون **S** أقل ما يمكن عندما $\frac{\partial S}{\partial ak}=0$

وسندخل الآن في مثال عملي على ذلك

الجدول أسفله يعطي بيانات عن مدى استهلاك الماء بإحدى البلدان وذلك ببلايين الجالونات في اليوم

- استعمل الانكفاء الأسى لاستهلاك الماء بدلالة الزمن
- استعمل ما حصلت عليه لحساب استهلاك الماء في سنة 1975 وقارن بما كان متوقعا وهو 449.7

السنة	1930	1940	1950	1960	1970
الاستهلاك	**110.5**	**136.43**	**202.7**	**322.9**	**411.2**

(جدول 5-1)

وسنطبق ما سبق على المعادلة التالية (الاسية)

$Y=ab^x$

ولكي تصبح خطية نأخذ لوغارثم الطرفين

$$\mathbf{Ln\ (y) = ln\ (ab^x)}$$
$$\mathbf{Ln(y) = ln(a) + x\ ln(b)}$$
$$\mathbf{Z = A + B\ x}$$

حيث **B=ln(b)** و **A=ln(a)** و **Z = ln(y)**

$$S=\sum_{i=0}^{n} d^2 \quad =\sum_{i=0}^{n}(A+Bx-z)^2$$

$$\frac{\partial S}{\partial A}=2\sum_{i=0}^{n}(A+Bx-Z)(1)=0$$

$$\frac{\partial S}{\partial B}=2\sum_{i=0}^{n}(A+Bx-Z)(x)=0$$

المعادلات العمودية

$$nA \quad + B\sum x_i = \sum z_i$$
$$A\sum x_i + B\sum x_i^2 = \sum z_i x_i$$

وبارجاع $z = \ln(y)$

$$nA + B\sum x_i = \sum \ln(y_i)$$
$$A\sum x_i + B\sum x_i^2 = \sum x_i \ln(yi)$$

$$\Delta = \begin{vmatrix} n & \sum xi \\ \sum xi & \sum xi^2 \end{vmatrix} = n\sum xi^2 - (\sum xi)^2 \text{ ....(1)}$$

$$A = \begin{vmatrix} \sum \ln(yi) & \sum xi \\ \sum xi\ln(yi) & \sum xi^2 \end{vmatrix} / \Delta = [\sum xi^2 \sum \ln(yi) - \sum xi \sum xi\ln(yi)] / \Delta \text{.....(2)}$$

$$B = \begin{vmatrix} n & \sum \ln(yi) \\ \sum xi & \sum xi\ln(yi) \end{vmatrix} / \Delta = [n\sum xi\ln(yi) - \sum \ln(yi) \sum xi] / \Delta \text{ ....(3)}$$

$$a = e^A \quad , \quad b = e^B$$

بالعودة للمعادلة الرئيسية والتعويض بقيم a و b

$$Y = a\,b^x \text{ ....(4)}$$

وطبقا للبيانات المعطاة والتي تبين مدى استهلاك الماء بإحدى البلدان ببلايين الجالونات.

السنة	1930	1940	1950	1960	1970
الاستهلاك	**110.5**	**136.43**	**202.7**	**322.9**	**411.2**

(جدول 5-2)

ولو أردنا إيجاد الحل يدويا فنكمل الجدول ونختصر السنوات كالتالي:

x	y	Ln(y)	X^2	X*ln(y)	X^2*ln(y)
30	110.5	4.7	900	141.15	4234.51
40	136.43	4.9	1600	196.63	7865.30
50	202.7	5.3	2500	265.59	13279.32
60	322.9	5.7	3600	346.64	20798.43

70	411.2	6.0	4900	421.34	29493.49
250	1183.73	26.6	13500	1371.35	75671.05

(جدول 5-3)

بالتعويض عن قيم المجاميع

$\Delta = 5*13500-(250)^2 = 5000$

$A= (26.6*13500-250*1371.35)/5000 =3.25$

$B = (5*1371.35-250*26.6)/5000 =0.041$

$a=e^{3.25} = 25.79$

$b=e^{0.041} = 1.04$

$Y=ab^x$

$Y=25.79*(1.04)^x$

$y(75)=25.79*(1.04)^{75} =501.8$

هذه الأرقام العشرية ليست دقيقة بل مقربة لرقم أو رقمين ولكن النتيجة النهائية مأخوذة من بيانات محسوبة جيدا.

الجدول بعد الملائمة

x	30	40	50	60	70
y	104.36	147.94	209.72	297.31	421.46

(جدول 5-4)

قائمة بالمتغيرات المستخدمة في الخوارزمية والبرنامج

I: متغير للتكرار

N: عدد النقاط المدخلة

Xi,yi : قيم **x**(السنة) و**y**(الاستهلاك) المعطاة

x: السنة المراد ايجاد قيمة الاستهلاك عندها

Y: قيمة الاستهلاك الناتج عند **x**

delta: قيمة المحدد العام الناتج من المعادلات السابقة...... المعادلة (1)

sumx: مجموع قيم **x**

sumy: مجموع قيم **y**

sumxx: مجموع قيم مربع **x** أي مجموع x^2

Sumxy: مجموع قيم **x*y**

a: قيمة **a** معامل المعادلة الرئيسية **(2)**

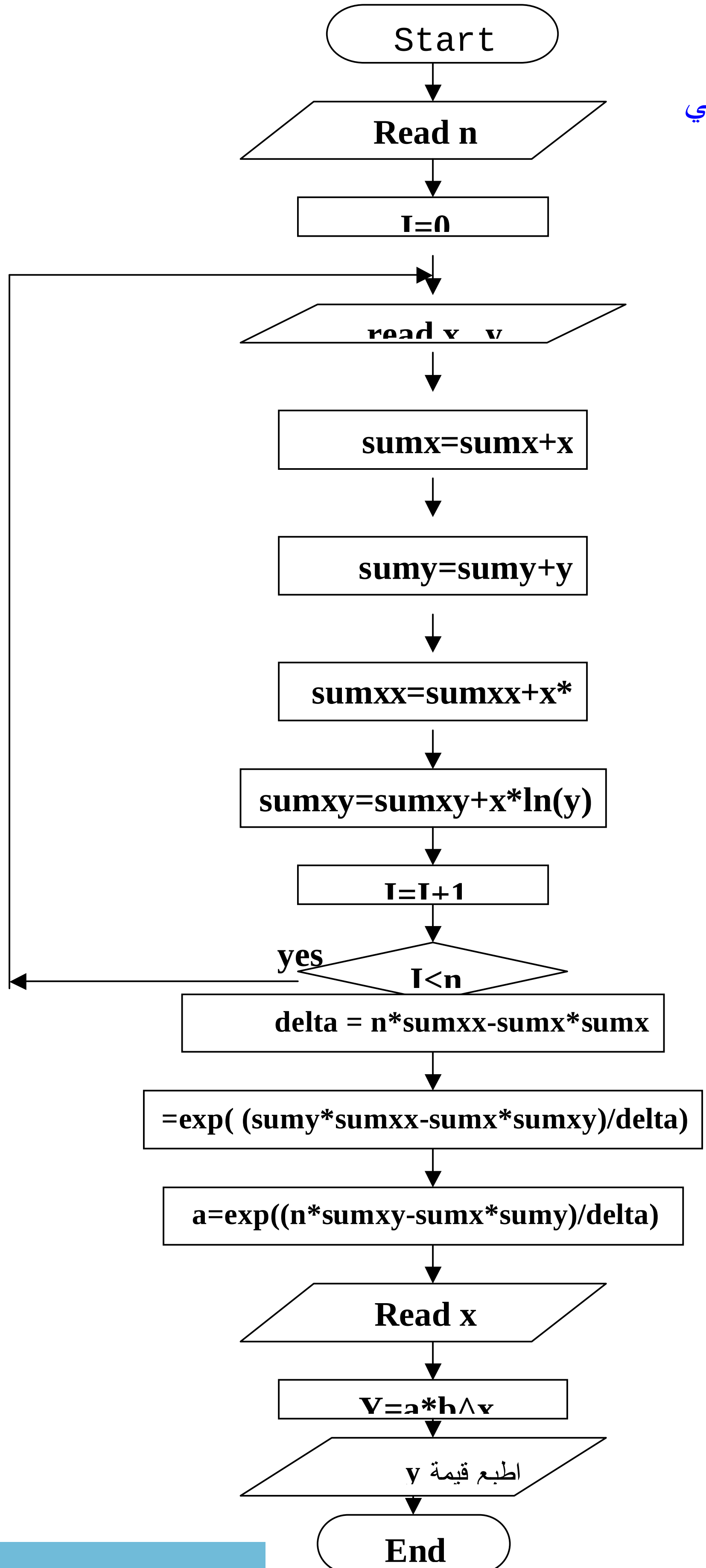
b: قيمة **b** معامل المعادلة الرئيسية **(3)**

الخوارزمية:

57. أبدا
58. ادخل عدد النقاط **n**
59. اجعل **I=0**
60. ادخل قيم **xi** و **yi** اللتان تمثلان السنة والاستهلاك على التوالي
61. احسب مجموع **xi** **(sumx=sumx+xi)**
62. احسب مجموع **ln(yi)** **(sumy=sumy+ln(yi))**
63. احسب مجموع **xi2** **(sumxx=sumxx+xi2)**
64. احسب مجموع **xi*ln(yi)** **(sumxy=sumxy+xi*ln(yi))**
65. احسب **I=I+1**
66. هل **I<n** إذا كان نعم ارجع إلى 4
67. احسب قيمة **delta** من (1) **a** من المعادلة (2) وقيمة **b** من المعادلة (3)
68. ادخل قيمة **x** المراد ايجاد **y** عندها
69. احسب قيمة **y** من المعادلة (4)
70. النهاية

المخطط الانسيابي

Flow Chart



(شكل 5-1)

البرنامج

```
#include<stdio.h>
#include<math.h>
void main()
{
int i,n,sumx,x,sumxx,xi[5];
double a,b,sumy,y,sumxy,delta,yi[5];

printf("Enter Points Number: ");
scanf("%d",&n);
printf("Enter Values of x and y ");
sumx=0;sumy=0;sumxy=0;sumxx=0;

for(i=0;i<n;i++)
{
fscanf("%d",&xi[i]);
scanf("%lf",&yi[i]);
```

```
 yi[i]=log(yi[i]);
 sumx=sumx+xi[i];
 sumy=sumy+yi[i];
 sumxx=sumxx+xi[i]*xi[i];
 sumxy=sumxy+xi[i]*yi[i];
 }
 delta = n*sumxx-sumx*sumx ;
 a=exp( (sumy*sumxx-sumx*sumxy)/delta);
 b=exp( (n*sumxy-sumx*sumy)/delta );

 printf("Enter value of x");
 scanf("%d",&x);
 y=a*pow(b,x);
 // writing
 printf("a= %lf , b= %lf\n\n",a,b);
 printf(" y = a * b ^ x\n\n");
 printf("x :");for(i=0;i<n;i++) printf("%d\t",xi[i] );
 printf("\ny :");for(i=0;i<n;i++)printf("%5.2lf
 ",a*pow(b,xi[i]));
 printf("\n\n y = %5.2lf * %5.2lf^%d = %lf\n\n",a,b,x,y);
 printf("When x=%d then y=%5.3lf",x,y);
 }
```

البرنامج: في صورته وهو يقرأ من ملف ويطبع إلى ملف

```
#include<stdio.h>
#include<math.h>
void main()
{
int i,n,sumx,x,sumxx,xi[5];
double a,b,sumy,y,sumxy,delta,yi[5];
 FILE *stream;
//  input values
stream = fopen("fitread.FIL", "r");
fscanf(stream,"%d",&n);
for(i=0;i<n;i++)   fscanf(stream,"%d",&xi[i]);
for(i=0;i<n;i++)   fscanf(stream,"%lf",&yi[i]);
fclose(stream);
// Calculation
sumx=0;sumy=0;sumxy=0;sumxx=0;
for(i=0;i<n;i++){yi[i]=log(yi[i]);
                 sumx=sumx+xi[i];
                 sumy=sumy+yi[i];
```

```
                  sumxx=sumxx+xi[i]*xi[i];
                  sumxy=sumxy+xi[i]*yi[i];
                 }
delta = n*sumxx-sumx*sumx ;
a=exp( (sumy*sumxx-sumx*sumxy)/delta);
b=exp( (n*sumxy-sumx*sumy)/delta );
printf("Enter value of x ");
scanf("%d",&x);
y=a*pow(b,x);
// writing
stream = fopen("fitwrite.FIL", "w+");
fprintf(stream,"a= %lf , b= %lf\n\n",a,b);
fprintf(stream," y = a * b ^ x\n\n");
fprintf(stream,"x :");
for(i=0;i<n;i++) fprintf(stream,"%d\t",xi[i] );
fprintf(stream,"\ny :");
for(i=0;i<n;i++) fprintf(stream,"%5.2lf  ",a*pow(b,xi[i] ));
fprintf(stream,"\n\n y = %5.2lf * %5.2lf^%d =
%lf\n\n",a,b,x,y);
fprintf(stream,"When x=%d then y=%5.3lf",x,y);
fclose(stream);
}
```

الإدخالات

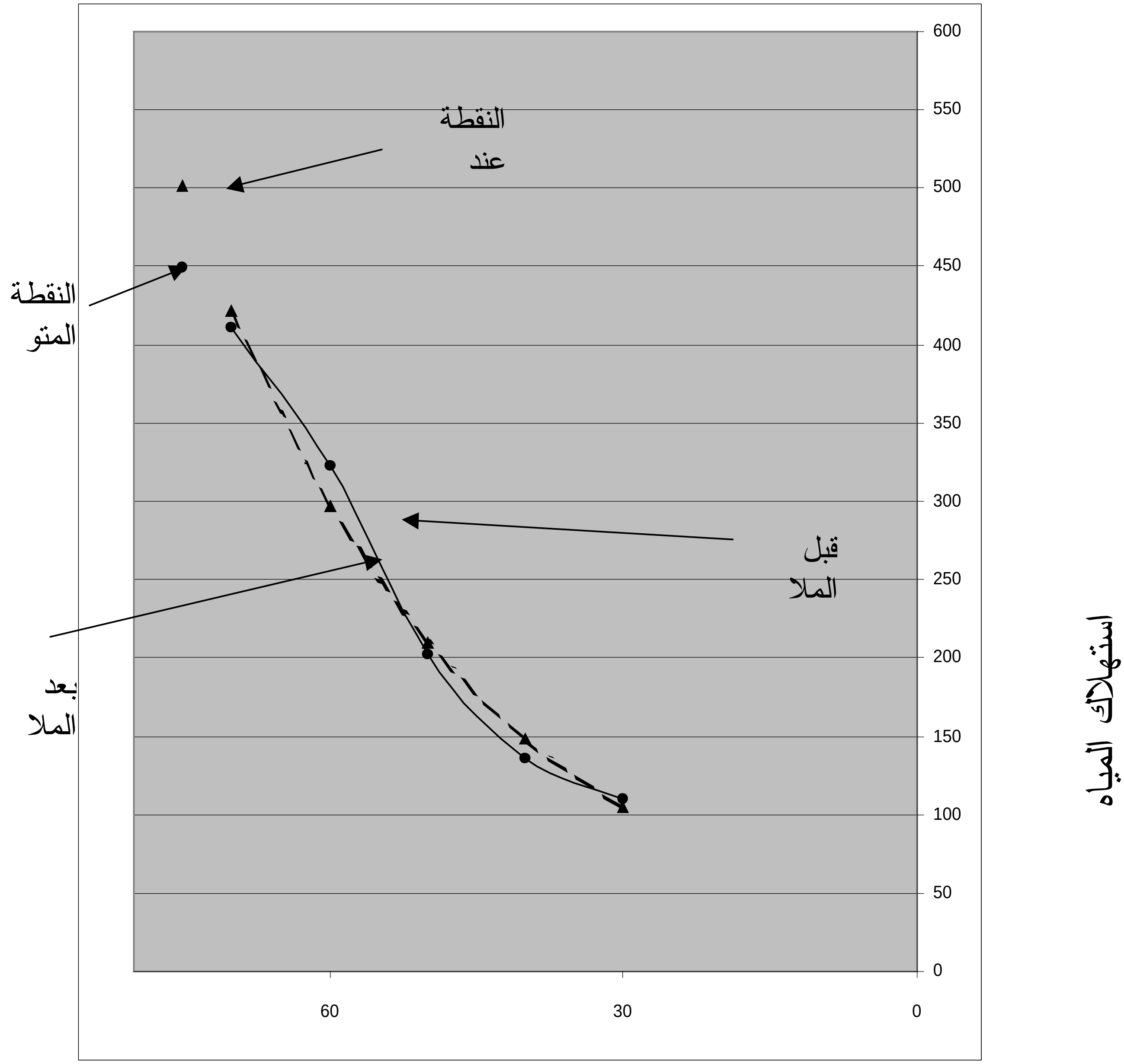
```
Enter Points Number: 5
Enter Value of x 30 40 50 60 70
Enter Value of y  110.5 136.43 202.7 322.9 411.2
Enter value of x 75
```

النتائج

```
a= 36.633592 , b= 1.035513

 y = a * b ^ x

x :  30      40       50       60       70
y :104.36  147.94  209.72  297.31  421.46

 y = 36.63 *  1.04^75 = 501.804287

When x=75 then y=501.804
```

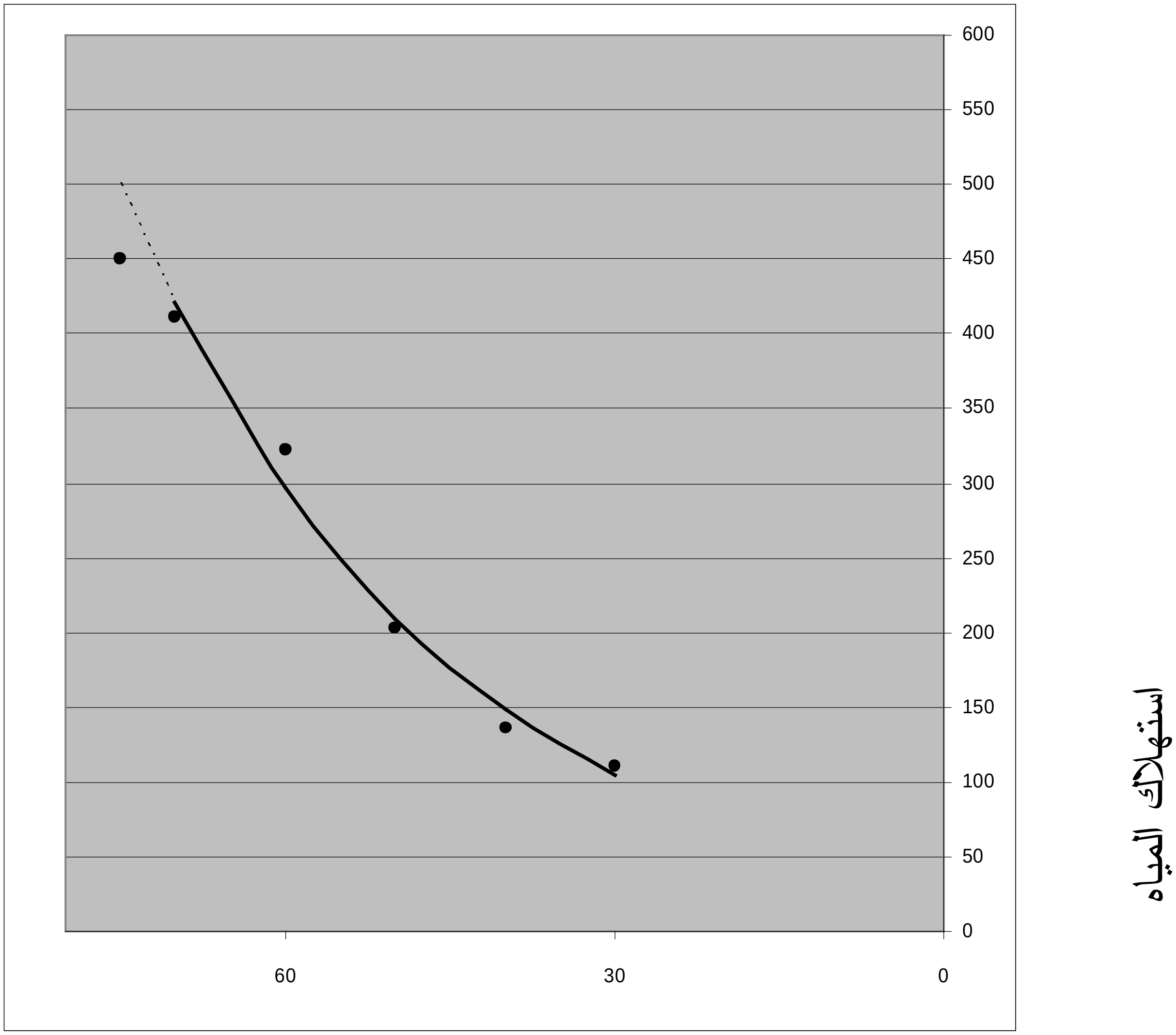
نلاحــظ أن المنحنى الملائم أخذ طريقا بين النقاط المعطاة ليلائم بينها ويقلل الخطأ الناتج أثناء القراءة

النقطة المتوقعة(449.7 بليون جالون) كانت بعيدة عن الناتجة في عام 1975 وقد أعطتنا الطريقة قيمة أقرب للصحة(501.8) مما يوفر لمصلحة المياه في تلك البلد رؤية عن استهلاكها في ذلك العام لتتخذ ما يلزم.



السنــــــــــــة

(شكل 5-2)



السنـــــــــــة

النقاط المعطاة مع منحنى الملائمة

(شكل5-3)

الخاتمة

في الختام لا يسعنا الا أن نسأل

الله عز وجل تحقيق الفائدة من هذا

الكتاب ونتمنى أنه لقي استحسانكم .

ويسرنا أن نتلقى آراءكم حول فحوى

الكتاب على عنوان البريد الالكتروني :

tkne@tkne.net

ولكم جزيل الشكر

عمر التومي   أحمد حمر الشوشة

وادارة موقع التقنية