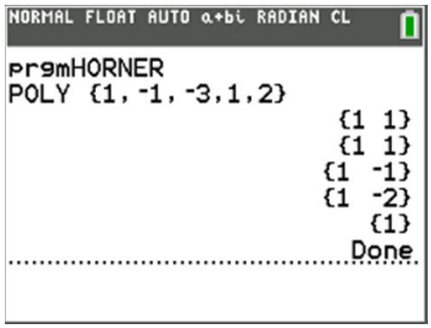
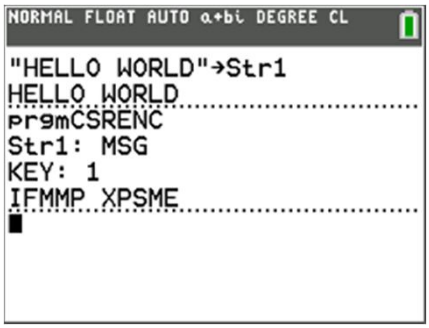
Cahiers T³ Europe
Vlaanderen nr. 45

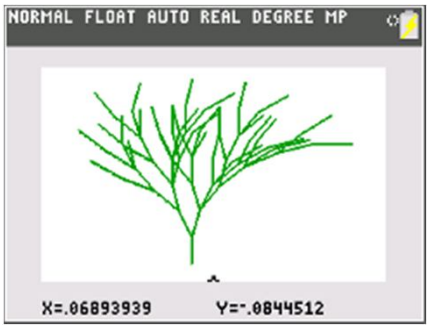
Onderzoekscompetenties: Wiskunde en moderne technologieën

Een verkenning van het gebruik van wiskunde en ICT in de hedendaagse wetenschap.

Didier Deses







KATHOLIEKE UNIVERSITEIT LEUVEN

Onderzoekscompetenties: Wiskunde en moderne technologieën

Dr Didier Deses[1]

[1]Leerkracht wiskunde KA Koekelberg, medewerker aan het departement wiskunde van de VUB, stuurgroep T^3-Vlaanderen

Samenvatting

In dit cahier gaan we een aantal onderwerpen aanraken die geschikt zijn als onderzoekscompetenties wiskunde/wetenschappen in het ASO. Elk hoofdstuk kan afzonderlijk door één leerling of in kleine groepjes doorgenomen worden. De onderwerpen komen uit de kruisbestuiving van wiskunde en technologie. In de eerste twee hoofdstukken wordt nagegaan hoe technologie kan helpen om wiskundige problemen aan te pakken. De volgende hoofdstukken gaan dieper in op een aantal voorbeelden waarbij de wiskunde de technologie ter hulp snelt.

Inhoudsopgave

Hoofdstuk 1

Numerieke wiskunde

De wiskunde bevat een aantal problemen waarvan bewezen is dat ze niet oplosbaar zijn, zoals bijvoorbeeld het exact berekenen van een nulwaarde van een veelterm van hoge graad (> 4) of het exact berekenen van oppervlakten onder bepaalde krommen (denk aan de normale verdeling). De numerieke wiskunde biedt een aantal benaderingsmethoden die middels de nodige technologie vaak resulteren in zeer bruikbare benaderingen. We behandelen hier enkele voorbeelden, gebaseerd op de leerstof van het ASO.

1.1 Benaderen van nulwaarden

Het bepalen van de nulwaarden van een functie kan soms zeer moeilijk zijn.

Opdracht 1. Voor veeltermen van de eerste en de tweede graad heb je uitgebreid geleerd om de nulwaarden of een ontbinding te zoeken. Voor veeltermen van hogere graden is dit veel moeilijker. Je hebt zeker voorbeelden gezien waar het ontbinden in factoren en de methode van Horner hulp kunnen bieden. Voor veeltermen van de derde en vierde graad bestaan er formules voor een "veralgemeende" discriminantmethode. Voor veeltermen van graad vijf of hoger is bewezen dat er geen algemene methode of formule bestaat om deze te ontbinden in factoren! Dit bewijs hebben we te danken aan Niels Henrik Abel. Zoek informatie op over deze wiskundige en over de stelling die hij bewees.

Opdracht 2. Geef een overzicht van de verschillende gevallen voor de nulwaarde van een eerstegraadsfunctie.

Opdracht 3. Geef een overzicht van de verschillende gevallen voor de nulwaarden van een tweedegraadsfunctie.

Opdracht 4. Geef een overzicht van de verschillende gevallen voor de nulwaarde van een derdegraadsfunctie. (**Hint:** gebruik de zoektermen *cubic function, cubic equation, cubic discriminant.*)

Gelukkig bestaan er numerieke methoden die ervoor zorgen dat men nulwaarden kan benaderen. We geven hier drie zulke methoden.

1.1.1 Bisectiemethode

In de les wiskunde heb je zeker gesproken over continuïteit en waarschijnlijk heb je onderstaande stelling gezien.

Stelling 1 (*Middelwaardestelling).* Zij $f : \mathbb{R} \to \mathbb{R}$ een functie, continu op het interval $]a, b[$ en rechtscontinu in a, linkscontinu in b. Indien $f(a) < f(b)$ (resp. $f(b) < f(a)$) dan bestaat er voor elke $c \in [f(a), f(b)]$ (resp. $c \in [f(b), f(a)]$) een $x \in [a, b]$ zodat $c = f(x)$.

Opdracht 5. Illustreer de middelwaardestelling aan de hand van een grafiek.

Een uiterst nuttige stelling die onmiddellijk uit de vorige volgt is de volgende:

Stelling 2 (*Bolzano).* Zij $f : \mathbb{R} \to \mathbb{R}$ een functie, continu op het interval $]a, b[$ en rechtscontinu in a, linkscontinu in b. Indien $f(a)$ en $f(b)$ een verschillend teken hebben, dan bezit f een nulwaarde in $[a, b]$.

Opdracht 6. Illustreer deze stelling door middel van een grafiek.

Het nut van deze stelling ligt vooral in haar vele toepassingen. Men kan hiermee gemakkelijk tonen dat elke derdegraadsveelterm (of veelterm van oneven graad) altijd minstens één nulwaarde heeft, gewoon door twee waarden uit te rekenen.

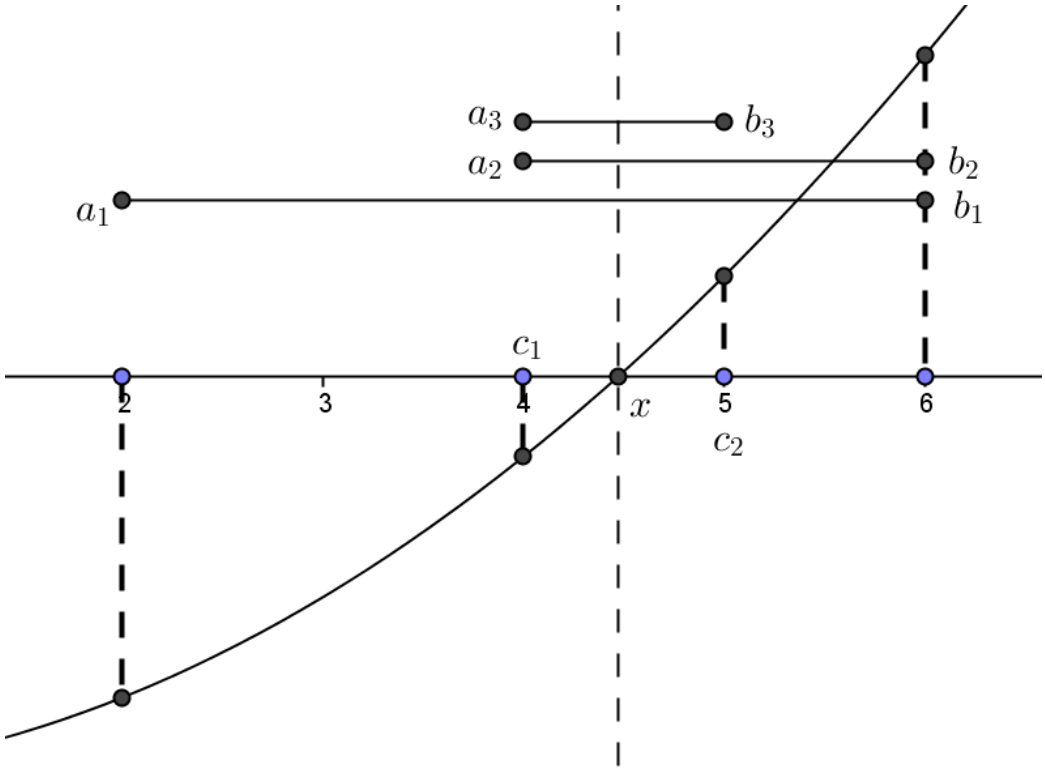
Opdracht 7. Gebruik de stelling van Bolzano om te bewijzen dat $f(x) = x^5 + x^4 + x^3 + 3x^2 - 2x + 5$ een nulwaarde heeft.

Opdracht 8. Gebruik de stelling van Bolzano om te tonen dat elke veelterm van oneven graad tenminste één nulwaarde bezit.

Een andere toepassing is het numeriek benaderen van nulwaarden door deze stelling iteratief toe te passen. Zij $f : \mathbb{R} \to \mathbb{R}$ een functie, continu op het interval $]a_1, b_1[$ en rechtscontinu in a_1, linkscontinu in b_1. Stel dat $f(a_1)$ en $f(b_1)$ een verschillend teken hebben. De functie heeft een nulwaarde x in $[a_1, b_1]$. Bepaal nu het midden $c_1 = \frac{a_1+b_1}{2}$. Als $f(c_1) = 0$ hebben we de nulwaarde gevonden. Indien $f(c_1)$ en $f(a_1)$ een verschillend teken hebben dan

stellen we $a_2 = a_1$ en $b_2 = c_1$ en anders, als $f(c_1)$ en $f(b_1)$ een verschillend teken hebben, kiezen we $a_2 = c_1$ en $b_2 = b_1$ (zie figuur). De functie heeft nu een nulwaarde $x \in [a_2, b_2] \subset [a_1, b_1]$. We berekenen opnieuw het midden $c_2 = \frac{a_2+b_2}{2}$ en herhalen de procedure. Aldus bekomen we een rij intervallen die steeds de nulwaarde x beter benaderen:

$$x \in \ldots \subset [a_3, b_3] \subset [a_2, b_2] \subset [a_1, b_1]$$

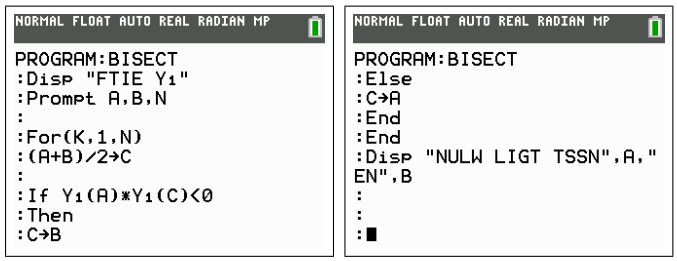


Opdracht 9.
Gebruik de **TI-84 Plus Color** om aan de hand van 2nd [calc] [zeros] de nulwaarde(n) te zoeken van x^2 en van $(x-1)(x+1)^2$. Wat merk je? Leg het verband met de bovenstaande methode.

Deze methode om nulpunten te benaderen (soms ook dichotomie of bisectiemethode genoemd) werkt enkel onder voorwaarde dat f van teken verandert in het nulpunt. De bisectiemethode kan bovendien op eenvoudige manier geprogrammeerd worden.

Het programma dat we geven gaat ervan uit dat de functie in y_1 zit. Er wordt gevraagd naar het interval $[a, b]$ waarin een nulwaarde zit en waarbij $y_1(a)$ en $y_1(b)$ een verschillend teken hebben. Bovendien wordt ook het aantal

iteraties n gevraagd. In het programma moeten we ondermeer nagaan of $y_1(a)$ en $y_1(c)$ een verschillend teken hebben, dit kan het gemakkelijkst door de voorwaarde $y_1(a) \cdot y_1(c) \leq 0$ na te gaan. Afhangend hiervan passen we de grenzen van het interval waarin de nulwaarde zit aan.



Opdracht 10. Bespreek het programma regel per regel of maak een stroomdiagram voor het algoritme van de bisectiemethode. Probeer het geheel zo eenvoudig mogelijk te maken.

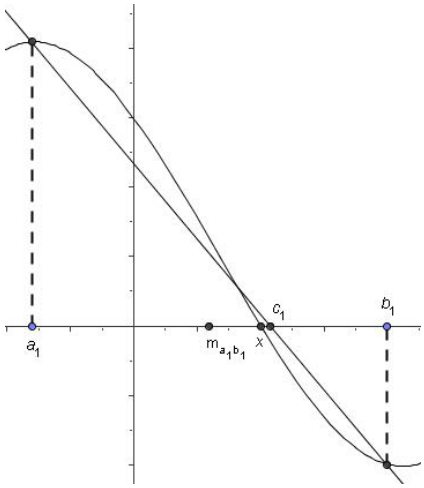
Indien de voorwaarden voldaan zijn (en we gaan er vanuit dat dit zo is, om het programma zo eenvoudig mogelijk te houden) zal het programma een betere benadering geven (kleiner interval) dan het interval waarvan men vertrekt. Indien de voorwaarden niet voldaan zijn, zal een foute benadering gegeven worden.

Opdracht 11. Door enkele extra regels toe te voegen kan men een foutief antwoord vervangen door een passende foutmelding, zoals de **TI-84 Plus Color** dat doet. Werk dit uit.

Opdracht 12. Werk een aantal voorbeelden uit, waarbij je $\sqrt{2}$ en de gulden snede benadert.

1.1.2 Regula falsi

De regula falsi methode is in essentie dezelfde als de bisectiemethode. Het enige verschil is de manier waarop de nieuwe grens c_1 wordt bepaald. In vele gevallen kan men een betere benadering geven dan het midden $m_{a_1 b_1} = \frac{a_1+b_1}{2}$, door volgende methode. We stellen c_1 gelijk aan de nulwaarde van de rechte door $(a_1, f(a_1))$ en $(b_1, f(b_1))$ (zie figuur).



We moeten nu echter een formule vinden om de waarde van c_1 te bepalen. Hiervoor vertrekken we van de vergelijking van een rechte door twee punten.

$$y = \frac{y_2 - y_1}{x_2 - x_1}(x - x_1) + y_1$$

Toegepast op de punten $(a_1, f(a_1))$ en $(b_1, f(b_1))$ geeft dit de vergelijking:

$$y = \frac{f(b_1) - f(a_1)}{b_1 - a_1}(x - a_1) + f(a_1)$$

Het vinden van de nulwaarde betekent dus het oplossen naar x van de vergelijking:

$$0 = \frac{f(b_1) - f(a_1)}{b_1 - a_1}(x - a_1) + f(a_1)$$

Dit geeft:

$$\begin{aligned}
\iff \qquad 0 &= \tfrac{f(b_1)-f(a_1)}{b_1-a_1}x - \tfrac{f(b_1)-f(a_1)}{b_1-a_1}a_1 + f(a_1) \\
\iff \quad \tfrac{f(b_1)-f(a_1)}{b_1-a_1}x &= \tfrac{f(b_1)-f(a_1)}{b_1-a_1}a_1 - f(a_1) \\
\iff \qquad x &= \tfrac{b_1-a_1}{f(b_1)-f(a_1)}\left(\tfrac{f(b_1)-f(a_1)}{b_1-a_1}a_1 - f(a_1)\right) \\
\iff \qquad x &= a_1 - \tfrac{b_1-a_1}{f(b_1)-f(a_1)}f(a_1) \\
\iff \qquad x &= \tfrac{a_1(f(b_1)-f(a_1))-(b_1-a_1)f(a_1)}{f(b_1)-f(a_1)} \\
\iff \qquad x &= \tfrac{a_1f(b_1)-b_1f(a_1)}{f(b_1)-f(a_1)}
\end{aligned}$$

We moeten dus in het algoritme van de bisectiemethode c_1 bepalen door:

$$c_1 = \frac{a_1 f(b_1) - b_1 f(a_1)}{f(b_1) - f(a_1)}$$

Iteratie levert dan opnieuw een rij intervallen die het nulpunt steeds beter benaderen en dit op een manier die in het algemeen betere benaderingen oplevert dan de bisectiemethode.

Opdracht 13. Maak een stroomdiagram voor een programma dat de regula falsi gebruikt.

Opdracht 14. Schrijf een programma dat de regula falsi gebruikt om nulwaarden van een functie te benaderen.

Opdracht 15. Vergelijk de bisectiemethode en de regula falsi. Wat kan je concluderen?

Opdracht 16. Zoek op waarom deze methode de regula falsi heet.

1.1.3 Newton-Raphson

Opdracht 17. Bepaal de vergelijking van de raaklijn aan de kromme $y = x^2 - 2$ in het punt $a = 1$. Maak een tekening. Benader het nulpunt $\sqrt{2}$ door de nulwaarde van de raaklijn.

Bovenstaande methode om een nulpunt te benaderen kan door iteratie uitgebreid worden. Beschouw een functie f en een benadering x_1 van haar nulwaarde. In x_1 is de raaklijn gegeven door

$$y = f'(x_1)(x - x_1) + f(x_1)$$

Het snijpunt van de raaklijn met de xs is een oplossing van

$$0 = f'(x_1)(x - x_1) + f(x_1)$$

m.a.w.

$$x = x_1 - \frac{f(x_1)}{f'(x_1)} \text{ (Reken dit na!)}$$

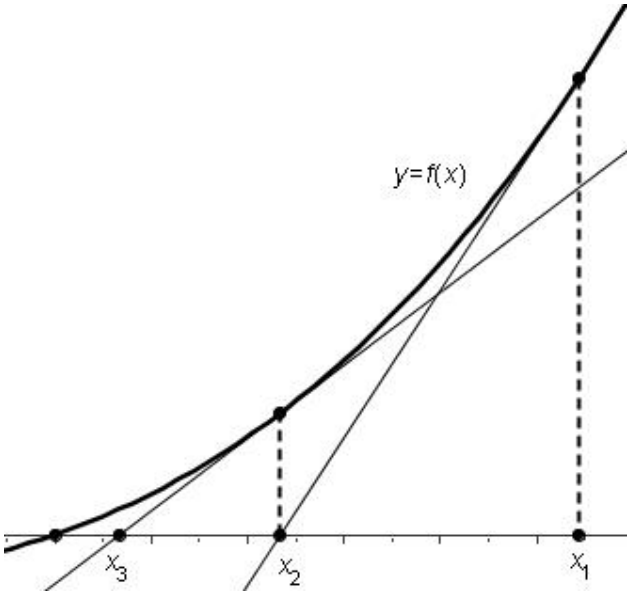
Dit geeft meestal een betere benadering van de nulwaarde van f

$$x_2 = x_1 - \frac{f(x_1)}{f'(x_1)}$$

Opnieuw nemen we nu de raaklijn in x_2 en bepalen we haar nulwaarde. Algemeen geeft

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

telkens betere benaderingen voor de nulwaarde van f.



Deze methode wordt de methode van Newton-Raphson genoemd.

Opdracht 18. Pas de methode toe van Newton-Raphson om het voorbeeld verder uit te werken en $\sqrt{2}$ nog beter te benaderen door een breuk.

Opdracht 19. Gebruik het commando `math` `[nDeriv]` om de methode van Newton-Raphson te programmeren op de **TI-84 Plus Color** .

Opdracht 20. Pas de methode toe om $\sqrt{2}$ te benaderen aan de hand van de functie $y = x^2 - 2$. Wat merk je als de beginwaarde 0 wordt gekozen? Kan je dit verklaren?

Opdracht 21. Vergelijk de methode van Newton-Raphson met de vorige methoden.

In de Babylonische cultuur (2000 VC) kende men reeds de waarde van $\sqrt{2}$. Maar hoe rekende men toen zulke wortels uit? Men gebruikte het algoritme dat nu bekend staat onder "guess, divide and average". Om $\sqrt{a}$ te benaderen doet men drie stappen:

1. doe eerst een gok naar de waarde van $\sqrt{a}$
2. deel n door je gok
3. maak het gemiddelde van 1. en 2.

De verkregen waarde is een betere benadering van $\sqrt{a}$ dan de gok waarvan je vertrok. Je kan nu de drie stappen herhalen met deze nieuwe waarde on een nog betere waarde te vinden.

Op de **TI-84 Plus Color** kan dit snel d.m.v. [Ans]. Tik eerst je gok in en druk op [enter]. Daarna geef je `(Ans+a/Ans)/2` in (waarbij je de passende waarde voor `a` invult). Elke keer dat je nu op [enter] drukt krijg je een betere benadering.

Opdracht 22. Doe dit eens voor $\sqrt{3}$ en $\sqrt{2}$. Kan je hiermee ook de gulden snede benaderen?

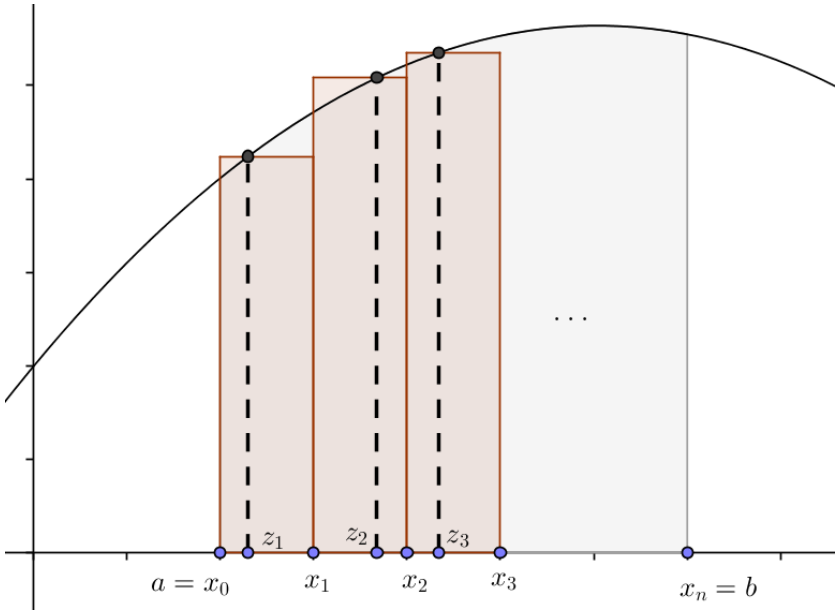
Opdracht 23. Waarom werkt deze methode? $\sqrt{a}$ is een nulpunt van $f(x) = x^2 - a$. Pas nu op $f(x)$ de methode van Newton-Raphson toe en bewijs dat dit het bovenstaande algoritme geeft.

1.2 Numerieke integratie

De oppervlakte onder een kromme $y = f(x)$ op het interval $[a, b]$ wordt gegeven door de bepaalde integraal. Deze wordt gedefinieerd als de limiet van een Riemann-som:

$$\int_a^b f(x)\ \mathrm{d}x = \lim_{n \to +\infty} \sum_{i=1}^{n} f(z_i)\Delta x$$

Waarbij het interval $[a, b]$ in n gelijke deelintervallen $[x_{i-1}, x_i]$ met lengte Δx wordt verdeeld en waarbij in het $i$de deelinterval een willekeurig punt z_i werd gekozen ($i = 1, 2, \ldots, n$).



De oppervlakte onder de kromme kan dus benaderd worden door een Riemannsom:

$$\int_a^b f(x)\ \mathrm{d}x \approx \sum_{i=1}^{n} f(z_i)\Delta x$$

Waarbij we een keuze hebben om z_i te kiezen binnen het interval $[x_{i-1}, x_i]$. Afhankelijk van de keuze die we nemen, bekomen we verschillende numerieke integratiemethoden.

1.2.1 Ondersom en bovensom

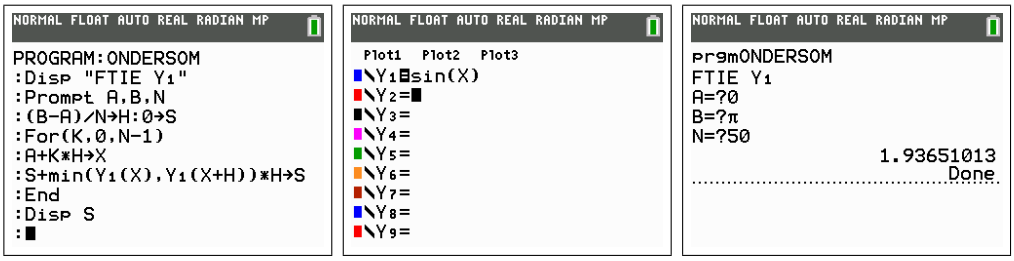
We benaderen nu een integraal als volgt. Het interval $[a, b]$ verdelen we in n kleinere intervallen $[x_{i-1}, x_i], i = 1, 2, \ldots, n$ telkens met eenzelfde breedte $\Delta x = h = \frac{b-a}{n}$. De benadering is dan

$$\int_a^b f(x)\ \mathrm{d}x \approx \sum_{i=1}^{n} f(z_i)h$$

door z_i zo te kiezen dat $f(z_i)$ de minimale waarde van f bereikt op $[x_{i-1}, x_i]$. De som zal dus altijd een waarde hebben die kleiner is dan de werkelijke oppervlakte, we noemen dit de ondersom.

Opdracht 24. Maak een passende tekening die de ondersom illustreert.

Volgend programma voert de ondersom-benadering uit.



Opdracht 25. Bespreek lijn per lijn wat het programma doet of maak een stroomdiagram.

Opdracht 26. Bij de beschrijving van de ondersom staat dat we z_i kiezen zodat $f(z_i)$ de minimale waarde van f bereikt op $[x_{i-1}, x_i]$. In het programma echter bekijken we enkel $\min\{f(x_{i-1}), f(x_i)\}$. Leg uit waarom dit equivalent is voor grote waarden van n.

Je kan ook de integraal benaderen door

$$\int_a^b f(x)\ \mathrm{d}x \approx \sum_{i=1}^{n} f(z_i)h$$

waarbij z_i zo wordt gekozen dat $f(z_i)$ de maximale waarde van f is op $[x_{i-1}, x_i]$. De som zal dus altijd een waarde hebben die groter is dan de werkelijke oppervlakte, we noemen dit de bovensom.

Opdracht 27. Maak een nieuw programma door het bovenstaande aan te passen om een bovensom uit te rekenen.
Opmerking: Het is niet nodig om alles opnieuw in te geven. Je kan in de programma-editor de recall-functie gebruiken `2nd` `[rcl]` gevolgd door `program` `[exec]` en dan het oorspronkelijke programma. Hierna wordt de volledige code gekopieerd in het nieuwe programma.

Opdracht 28. Gebruik bovenstaande programma's om een aantal bepaalde integralen te benaderen door onder- en bovensommen.

1.2.2 Midpuntsregel

Een betere manier van benaderen wordt als volgt gegeven. Het interval verdelen we opnieuw in n kleinere intervallen $[x_{i-1}, x_i], i = 1, 2, \ldots, n$ met een breedte $h = \frac{b-a}{n}$. Deze keer kiezen we z_i juist in het midden van het deelinterval te kiezen, d.w.z. $z_i = \frac{x_{i-1}+x_i}{2}$. De Riemann-som wordt dan (ga dit na):

$$\int_a^b f(x)\ \mathrm{d}x \simeq \sum_{i=1}^{n} f(\frac{x_{i-1}+x_i}{2})h$$

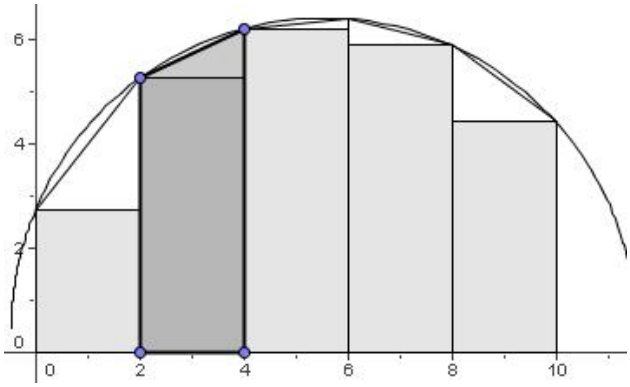
Opdracht 29. Maak een passende tekening die de midpuntsregel illustreert.

Opdracht 30. Schrijf een programma om aan de hand van de midpuntsregel een bepaalde integraal te benaderen.

Opdracht 31. Werk een aantal voorbeelden uit en vergelijk met de bovenstaande methodes.

1.2.3 Trapeziumregel

Tot nu toe hebben we de oppervlakte altijd benaderd door een onderverdeling aan de hand van rechthoeken. Het is echter goed mogelijk om andere vormen te gebruiken. De meest eenvoudige is het trapezium.



De oppervlakte van één trapezium wordt gegeven door $\frac{f(x_{i-1}) + f(x_i)}{2} h$ (Waarom?). We bekomen dus

$$\int_a^b f(x)\ \mathrm{d}x \simeq \sum_{i=1}^{n} (f(x_{i-1}) + f(x_i)) \frac{h}{2}$$

Opdracht 32. Schrijf een programma om aan de hand van de trapeziumregel een bepaalde integraal te benaderen.

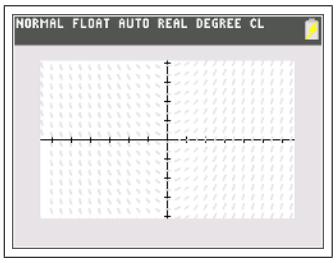
Opdracht 33. Werk een aantal voorbeelden uit en vergelijk met de bovenstaande methodes.

1.3 Differentiaalvergelijkingen

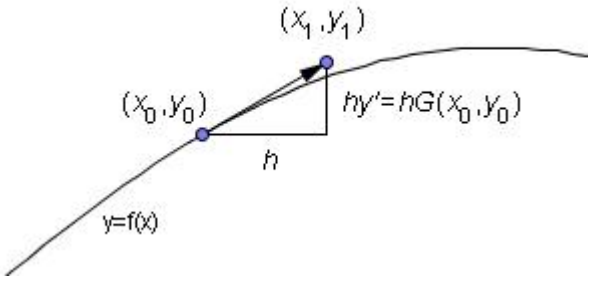
Vanaf het ogenblik dat men beschikt over het begrip afgeleiden kan men volgend probleem schetsen. Stel dat men van een bepaalde functie $y = f(x)$ informatie heeft over de afgeleide bijvoorbeeld $y' = G(x, y)$, kan men dan de

grafiek van f terugvinden? De hierbij vermelde vergelijking is een eenvoudig voorbeeld van een differentiaalvergelijking. Sommige differentiaalvergelijkingen kan men oplossen door te integreren, maar lang niet allemaal. Toch kan men op eenvoudige manier de grafiek van f terugvinden, zelfs op een **TI-84 Plus Color** . Dit steunt op de numerieke integratiemethode van Euler.

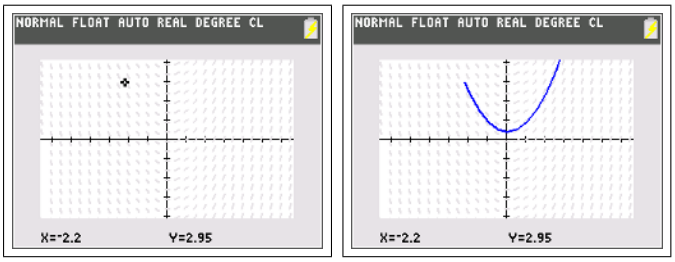
Wat eigenlijk gegeven is, is de afgeleide (dus een raakvector) in elk punt van het vlak want $y' = f'(x) = G(x, y)$. We kunnen dus in elk punt van het vlak een kleine raakvector tekenen. Men bekomt aldus een "fieldplot". Indien we het voorbeeld $y' = x$ gebruiken, ziet de fieldplot er als volgt uit:



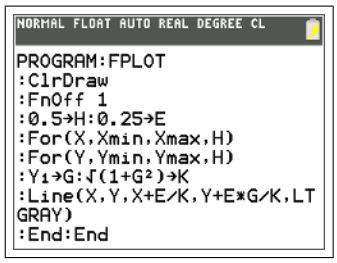
Hoe kunnen we nu hieruit de grafiek van f halen? Stel dat we van een zeker punt (x_0, y_0) veronderstellen dat het op de kromme $y = f(x)$ ligt. In dit punt kennen we een raakvector $(1, y') = (1, G(x_0, y_0))$. Als we nu een klein stapje $(h > 0)$ zetten in de x-richting en in de y-richting een stapje $hG(x_0, y_0)$, dan volgen we de raakvector en komen we in een punt $(x_1, y_1) = (x_0+h, y_0+hG(x_0, y_0))$ terecht dat zeer dicht bij de kromme ligt. In dit nieuwe punt kunnen we opnieuw beginnen en aldus volgen we stapje na stapje de kromme.



Wanneer we al deze punten tekenen, zien we een goede benadering voor de werkelijke kromme $y = f(x)$. Merk op dat we wel een keuze hebben. Het eerste punt kan je vrij kiezen, de kromme die dan berekend wordt, zal altijd een oplossing zijn, we noemen dit eerste punt de beginvoorwaarde. In ons voorbeeld blijkt de oplossing de vorm van een parabool te zijn. In dit geval hadden we de algemene oplossing $y = \frac{x^2}{2}+c$ ook kunnen vinden via integratie.



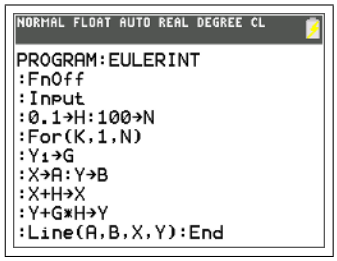
We keren nu naar de vraag hoe we dit alles in de **TI-84 Plus Color** krijgen. Hiervoor moeten we een beetje programmeren. Volgend programma maakt een fieldplot van de differentiaalvergelijking $y' = G(x, y)$.



- De eerste regel zorgt ervoor dat geen enkele functie uit y= getekend wordt, dit is nodig want we zullen y_1 gebruiken om de functie $G(x, y)$ in op te slaan.
- De tweede en de derde regel zorgen voor een standaardvenster zonder assenkruis (anders is het beeld niet meer overzichtelijk).
- We zullen nu het vlak onderverdelen in een raster van punten die in de x- en de y-richting op een afstand H van elkaar liggen. In elk van deze punten zullen we een (genormaliseerde) raakvector van lengte E tekenen.
- Na de definitie van H en E zorgen twee `for`-lussen ervoor dat elk punt in het raster wordt doorlopen. In elk van deze punten wordt via y_1 de functie $G(x, y)$ berekend. De waarde wordt opgeslagen in G.
- Vervolgens tekent men de genormaliseerde raakvector $(\frac{1}{\sqrt{1+G(x,y)^2}}, \frac{G(x,y)}{\sqrt{1+G(x,y)^2}})$ als een lijntje vanuit (x, y) met lengte E.
- Tenslotte worden beide `for`-lussen beëindigd.

Opdracht 34. Maak een stroomdiagram van het programma. Tracht dit zo overzichtelijk mogelijk te doen.

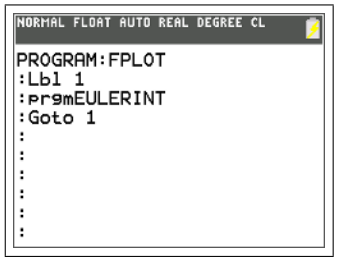
We schrijven nu een programma dat de Eulermethode gebruikt om een oplossing te tekenen van de differentiaalvergelijking $y' = G(x, y)$.



- Eerst wordt er opnieuw voor gezorgd dat de functies uit [y=] niet worden getekend, we zullen immers opnieuw y_1 gebruiken om de functie $G(x, y)$ in op te slaan. Het scherm wordt deze keer niet leeggemaakt d.m.v. `[zdecimal]` omdat we eventuele output van `FPLOT` willen blijven zien.
- Daarna gebruiken we [prgm] `[i/o]` `[input]` zonder bijkomend argument. De gebruiker krijgt dan het grafisch venster te zien en kan met de pijltjestoetsen de cursor bewegen en de gewenste x en y coördinaten kiezen.
- Vanuit het gekozen startpunt, gaan we telkens met een stap H verder, en het programma zal uiteindelijk N punten uitrekenen.
- Hierna wordt de `for`-lus gestart en wordt via y_1 de functie $G(x, y)$ berekend en in de variabele G gestoken.
- Alvorens het volgende punt te berekenen worden de coördinaten onthouden in A en B, zodat we later vanuit dit punt naar het nieuwe een lijnstukje kunnen tekenen. Dan worden de nieuwe x en y waarden uitgerekend, het volgende punt wordt dus bepaald.
- Uiteindelijk wordt het lijnstukje getekend en sluit de lus zich.

Opdracht 35. Maak een stroomdiagram van het programma. Tracht dit zo overzichtelijk mogelijk te doen.

We hebben dit programma zo geschreven dat we het kunnen laten aansluiten op het programma `FPLOT`. Doe dit door aan `FPLOT` volgende regels toe te voegen.



Opdracht 36. Gebruik de bovenstaande programma's om de differentiaalvergelijking $y' = -2xy$ op te lossen, bepaal ook de algemene oplossing. Doe hetzelfde voor $y' = e^{-x^2}$. Wat is hierbij het probleem? Is dit een probleem voor onze programma's op de **TI-84 Plus Color** ?

Opdracht 37. Het algoritme van Euler is niet echt bijzonder goed. Het vertoont zekere onstabiliteiten. Beschouw bijvoorbeeld eens $y' = \frac{-2}{y}$.

Opdracht 38. Wanneer de x-as de tijd voorstelt, zal een horizontale asymptoot overeenstemmen met een evenwicht op lange termijn. Beschouw de differentiaalvergelijking $y' = \sin 2y$. Hoe hangt het evenwicht af van de beginwaarde?

Hoofdstuk 2

Symbolische wiskunde

Binnen de wiskunde worden zeer vaak berekeningen gemaakt met symbolen zoals bijvoorbeeld het uitwerken of ontbinden in factoren van veeltermen. Sommige berekeningen zijn echter technisch van aard en duren lang. De moderne technologie is in staat om wiskundigen hierbij te helpen. In dit hoofdstuk bekijken we hoe een rekenmachine in staat is om abstracte symbolische berekeningen uit te voeren.

2.1 Veeltermen en hun bewerkingen

Opdracht 39. Formuleer de correcte definitie van een veelterm (of *polynoom*) van de $n$de graad. Zoek op wat de verzamelingen $\mathbb{R}[x], \mathbb{Q}[x]$ zijn.

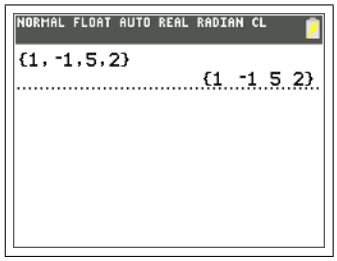
De essentie van de veelterm

$$V(x) = a_n x^n + \ldots + a_1 x + a_0$$

zit vervat in de lijst van de coëfficiënten

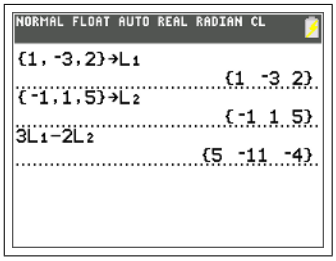
$$a_n, \ldots, a_1, a_0$$

We kunnen dus een veelterm weergeven aan de hand van een lijst. Dit kunnen we gebruiken om een veelterm in te geven op de **TI-84 Plus Color** . De veelterm $V(x) = x^3 - x^2 + 5x + 2$ wordt dan



Deze vorm van veeltermen gebruiken op een rekentoestel heeft slechts zin indien je er ook bewerkingen mee kan uitvoeren. Lijsten kan je optellen en vermenigvuldigen met een getal. Alsdus kunnen we lineaire combinaties van veeltermen berekenen. We werken volgend voorbeeld uit.

$$3(x^2 - 3x + 2) - 2(-x^2 + x + 5)$$



Wanneer veeltermen echter van graad verschillen komen we in de problemen.

Opdracht 40. Werk volgende bewerkingen uit aan de hand van de **TI-84 Plus Color** .

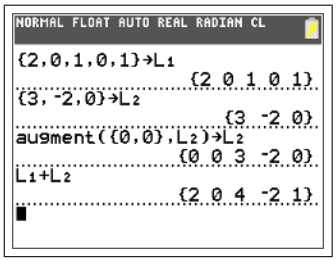
$$2(x^3 - 2x^2 + x + 2) + 5(-x^2 - 1)$$

Wat merk je? Hoe komt dit?

We kunnen bovenstaand probleem oplossen door termen met coëfficiënt 0 toe te voegen tot de graden gelijk zijn. Dit gebeurt door het commando 2nd`[list][ops][augment]`. Zo wordt

$$(2x^4 + x^2 + 1) + (3x^2 - 2x)$$

als volgt berekend



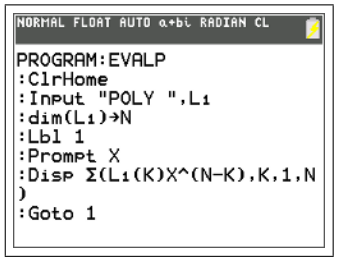
Opdracht 41. Reken nu opdracht 40 correct uit.

Opdracht 42. Indien L_1 een veelterm voorstelt, wat doen dan volgende commando's?

`augment(`L_1`,{0})`
`augment(`L_1`,{0,0})`

Naast het optellen en vermenigvuldigen willen we graag een veelterm in een punt evalueren.

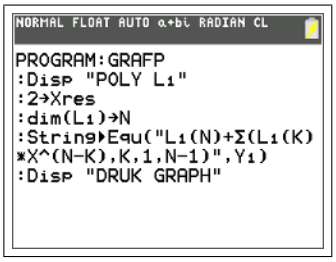
Opdracht 43. Bekijk volgend programma en bespreek regel per regel of maak een stroomdiagram. Σ vind je in `math` `[summation]`. Let op de volgorde van de getallen in L_1!

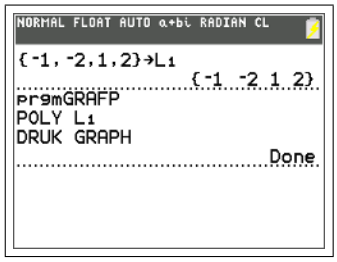


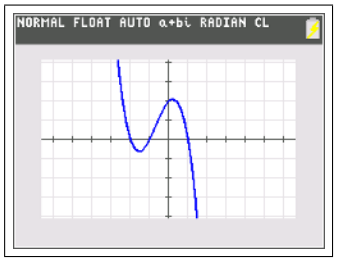
Opdracht 44. Wat gebeurt er wanneer we met bovenstaand programma een veelterm in $x = 0$ trachten uit te rekenen? Hoe komt dit?

Opdracht 45. Pas het programma aan zodat dit probleem opgelost wordt.

Je kan dezelfde techniek gebruiken om de grafiek van een veelterm te maken. Met `2nd` `[catalog]` `[String>Equ]` kan je de formule in Y_1 plaatsen zodat de grafiek gemaakt kan worden. Het berekenen van een functiewaarde vergt met deze methode wel extra tijd, vandaar dat `Xres` gelijk wordt gesteld aan 2, er zal dan maar om de twee pixels een functiewaarde berekend worden.





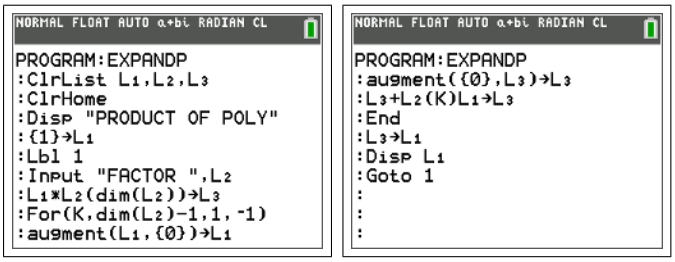


2.2 Uitwerken en ontbinden in factoren

Tot nu toe hebben we slechts een lineaire combinatie van polynomen symbolisch laten uitrekenen. We kunnen echter ook een product van polynomen laten uitrekenen. Veronderstel dat $V_1(x)$ en $V_2(x)$ veeltermen zijn en dat $V_2(x) = a_n x^n + \ldots + a_1 x + a_0$. Dan is het product

$$V_3(x) = V_2(x) \cdot V_1(x) = a_n \cdot x^n \cdot V_1(x) + \ldots + a_1 \cdot x \cdot V_1(x) + a_0 \cdot V_1(x)$$

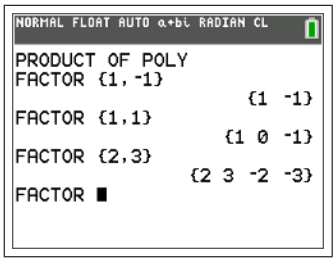
De termen $x^k \cdot V_1(x)$ kunnen zeer snel bekomen worden via `augment` (zie opdracht 42).



- Eerst worden de lijsten L_1, L_2 en L_3 leeg gemaakt. Het scherm wordt gewist en de titel afgedrukt.
- L_1 wordt gelijkgesteld aan 1, klaar om vermenigvuldigd te worden met een factor.
- Voor het ingeven van de eerste factor wordt een `label` geplaatst, zodat het programma later naar deze plaats kan terugkeren om een tweede factor op te vragen.
- De berekening van het product an e veelterm in L_1 met die in L_2 begint door L_1 te vermenigvuldigen met $L_2(\texttt{dim}(L_2))$, dit is het laatste element van L_2 (de constante term). We berekenen dus $a_0 \cdot V_1(x)$. Dit wordt tijdelijk in L_3 opgeslagen.
- Nu begint een `for`-lus die de elementen van L_2 in stijgende graad zal afgaan.
- L_1 wordt aangepast zodat men achtereenvolgens $x \cdot V_1(x), x^2 \cdot V_1(x), \ldots$ bekomt in de `for`-lus.
- De graad van het tussenresultaat L_3 wordt aangepast zodat men in de volgende stap dit tussenresultaat en $a_k \cdot x^k \cdot V_1(x)$ kan optellen.
- Na de `for`-lus staat het resultaat in L_3. Dit wordt overgeplaatst naar L_1 en afgedrukt.
- Het programma keert terug naar het `label` om een volgende factor op te vragen.
- Het programma stopt niet vanzelf. Om dit te doen druk je `on` en kies je `[Quit]`. Het laatste resultaat steekt in L_1, handig om nadien verder mee te werken.

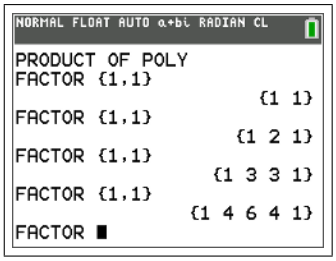
Opdracht 46. Maak een stroomdiagram van het programma. Tracht dit zo overzichtelijk mogelijk te doen.

Als voorbeeld berekenen we $(x-1)(x+1)(2x+3)$:



Opdracht 47. Gebruik het programma om voorbeelden van de verschillende merkwaardige producten uit te werken. Alsook de som- en productregel voor tweedegraadsveeltermen.

Opdracht 48. Wat wordt er hieronder uitgerekend? Leg het verband met de driehoek van Pascal en het Binomium van Newton. Wat zal de volgende regel zijn?



Opdracht 49. Reken uit en controleer:

$$(x^3 - x^2 + x - 1)(3x^3 + 2x^2 + x)$$

Opdracht 50. Maak een veelterm met een dubbele nulwaarde en twee enkele nulwaarden. Maak tevens de grafiek.

Nu we producten van veeltermen kunnen uitrekenen, richten we ons naar het ontbinden in factoren. De meest gebruikte methode voor veeltermen van hogere graad is de methode van Horner. In onderstaand voorbeeld gebruiken we deze methode om $V(x) = x^3 - 6x^2 + 11x - 6$ te ontbinden in $V(x) = (x-2)(x^2 - 4x + 3)$.

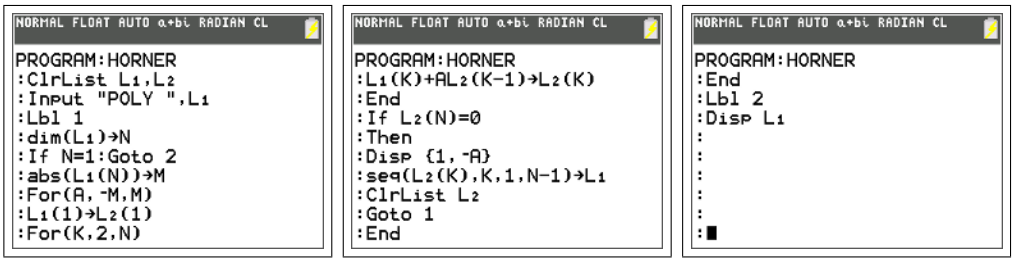
	1	-6	11	-6
2		2	-8	6
	1	-4	3	|0

We herhalen kort enkele feiten:

- De coëfficiënten van de gegeven veelterm $V(x)$ komen in de eerste rij te staan.
- Het Hornergetal (hier $a = 2$) komt helemaal links.
- Na het uitvoeren van de methode van Horner staat helemaal onderaan rechts de waarde van de veelterm in a. Hier is dit $V(a) = 0$. Dit betekent dat a een nulwaarde is.
- Indien $V(a) = 0$ dan is $V(x)$ deelbaar door $(x - a)$. De resterende factor (hier $x^2 - 4x + 3$) staat dan in de onderste regel.
- Wanneer a een geheel getal is, dan is dit een deler van de constante term uit $V(x)$. Om de gehele Hornergetallen te vinden kan men dus zoeken naar de gehele delers van deze constante term.

Opdracht 51. Maak een overzicht van de verbanden tussen volgende begrippen voor veeltermen: ontbinding in factoren, veeltermvergelijking, Hornergetal, nulwaarden.

De methode van Horner leidt tot volgend programma:

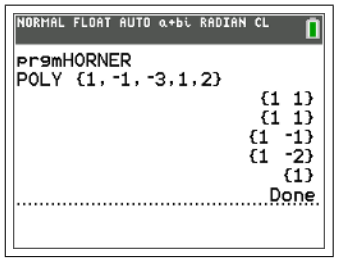


- Eerst worden de nodige lijsten gewist. Daarna wordt de veelterm opgevraagd en in L_1 gestoken.
- Indien de graad van de veelterm 1 is, kan er zeker geen ontbinding gevonden worden en stopt het programma.
- De constante term wordt bepaald en een `for`-lus wordt gestart om de mogelijke Hornergetallen uit te proberen.
- De hoogstegraadscoëfficiënt laat men zakken in het Hornerschema en aan de hand van een tweede `for`-lus wordt het schema verder uitgerekend. Het resultaat staat in L_2.

- Indien het laatste getal van L_2 nul is, weten we dat de factor $(x - a)$ voorkomt in de ontbinding. Deze wordt dan afgedrukt. De laatste rij van het Hornerschema komt nu in L_1 (in de juiste volgore van dalende machten). Hierna tracht het programma verder te ontbinden.
- Indien geen deler werd gevonden gaat het programma over naar een volgend Hornergetal.
- Indien tenslotte de `for`-lus geen verdere delers oplevert, stopt het programma. In L_1 staat de laatst overblijvende factor.

Opdracht 52. Maak een stroomdiagram van het programma. Tracht dit zo overzichtelijk mogelijk te doen.

Het programma kan je nu als volgt gebruiken.



Opdracht 53. Werk een aantal voorbeelden uit, die je eerst met `EXPANDP` hebt gemaakt.

Opdracht 54. Het programma zoekt de gehele wortels van een veelterm. Wat als er ook rationale wortels zijn? Beschouw volgende voorbeelden (werk ze ook met de hand uit).

1. $2x^3 - 2x^2 - 60x + 144$
2. $2x^3 - 3x^2 - 29x - 30$
3. $6x^3 + 19x^2 + 19x + 6$

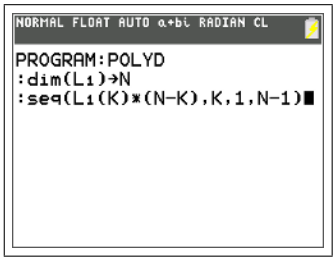
Opdracht 55. Indien het programma geen ontbinding vindt, betekent dit niet dat er geen ontbinding bestaat. Het kan ook zijn dat de nulwaarden niet geheel zijn. In het geval van een tweedegraadsveelterm biedt de discriminantmethode uitsluitsel. Schrijf een programma dat de ontbinding in factoren bepaalt d.m.v. de discriminantmethode.

Opdracht 56. Voor veeltermen van de eerste en de tweede graad heb je uitgebreid geleerd om de nulwaarden of een ontbinding te zoeken. Voor veeltermen van hogere graden is dit veel moeilijker. Je hebt zeker voorbeelden gezien waar het ontbinden in factoren en de methode van Horner hulp kunnen bieden. Voor veeltermen van de derde en vierde graad bestaan er formules voor een "veralgemeende" discriminantmethode. Voor veeltermen van graad vijf of hoger is bewezen dat er geen algemene methode of formule bestaat om deze te ontbinden in factoren! Dit bewijs hebben we te danken aan Niels Henrik Abel. Zoek informatie op over deze wiskundige en over de stelling die hij bewees.

2.3 Afgeleiden en primitieven

Nu we symbolisch veeltermen kunnen uitwerken en ontbinden in factoren (weliswaar in beperkte mate), zullen we trachten om afgeleiden en primitieven te laten uitrekenen.

Opdracht 57. Indien L_1 een veelterm voorstelt, wat geeft dan volgend programma?



Opdracht 58. Schrijf een programma dat een primitieve bepaalt van de veelterm in L_1.

Opdracht 59. Beschouw de veelterm

$$S(x) = \frac{1}{362880}\,x^9 - \frac{1}{5040}\,x^7 + \frac{1}{120}\,x^5 - \frac{1}{6}\,x^3 + x$$

1. Maak de grafiek van $S(x)$ op $[-\pi, \pi]$. Welke functie herken je?
2. Bepaal de afgeleide van $S(x)$ en maak de grafiek ervan. Stemt dit overeen met je antwoord op de vorige vraag? Verklaar.
3. Bepaal een primitieve van $S(x)$ en maak de grafiek ervan. Stemt dit overeen met je antwoord op de eerste vraag? Verklaar.

4. De stelling van Taylor zegt dat elke functie benaderd kan worden door een veelterm. Zoek deze stelling op en verklaar de notaties voor onderstaande veeltermen.

$$S(x) = \sum_{k=0}^{m} (-1)^k \frac{x^{2k+1}}{(2k+1)!}$$

$$C(x) = \sum_{k=0}^{m} (-1)^k \frac{x^{2k}}{(2k)!}$$

Het zijn trouwens deze twee veeltermen die in rekentoestellen en computers voorgeprogrammeerd zitten i.p.v. de echte goniometrische functies!

Hoofdstuk 3

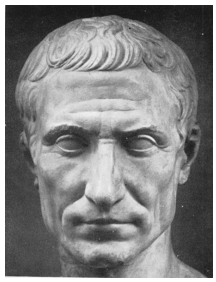
Beveiliging d.m.v. codes

In onze huidige technologische maatschappij is het noodzakelijk om informatie op vertrouwelijke wijzen door te sturen zonder dat iemand deze informatie kan achterhalen. Bovendien gebeurt het wel eens dat de informatie tijdens de overdracht verstoord wordt. Is het dan nog mogelijk om de exacte oorspronkelijke informatie terug te vinden? We behandelen hier enkele eenvoudige voorbeelden.

3.1 De Caesar-code

Reeds in de oudheid was het van belang om boodschappen door te sturen zonder dat de vijand de boodschap kon begrijpen indien deze toch onderschept werd. Julius Caesar maakte veelvuldig gebruik van de naar hem genoemde Caesar-code.

Opdracht 60. Zoek de nodige achtergrondinformatie over Julius Caesar.



Om een bericht te encoderen werd vooraf een sleutel afgesproken tussen Caesar en de bestemmeling. Deze sleutel K was een getal van 1 t.e.m. 25. Een bericht werd dan opgeschreven, waarna elke letter K plaatsen in het alfabet verschoven werd. Om het bericht te decoderen moest men gewoon elke letter van het alfabet in de andere zin verschuiven.

Opdracht 61. Om de Caesar-code te gebruiken was het nodig om een tabel te maken met het verschoven alfabet. Indien de afgesproken sleutel $K = 7$ is, vul dan onderstaande tabel aan.

alfabet	A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
gecodeerd	H ...

Opdracht 62. Gebruik bovenstaande tabel om volgend bericht te decoderen.

YVVKRHWQL LU KL DVSM.

Telkens er een andere sleutel wordt gebruikt, moet je bovenstaande tabel opnieuw maken. In de 15de eeuw ontwierp de Italiaanse Leone Battista Alberti een handig toestel om gemakkelijk de Caesar-code te kunnen gebruiken. Alberti was een typische renaissance-man: hij was schilder, dichter, taalkundige, filosoof, cryptograaf, musicus en architect. Zijn toestel noemde hij een *formula*, het bestond uit twee draaiende schijven.

Opdracht 63. Maak zelf een *formula*.

1. Knip een schijf van diameter 10cm en eentje van diameter 8cm uit een stuk papier of karton.
2. Beide schijven verdeel je in 26 sectoren.
3. Langs de omtrek zet je in elke sector een letter van het alfabet, op de grootste schijf zet je ook telkens de bijpassend sleutel van 0 t.e.m. 25.
4. Plaats beide schijven op elkaar en maak ze in het midden vast met een plooiduimspijker, zodat beiden kunnen draaien.

Door de ene schijf te draaien t.o.v. de andere kan je telkens een andere Caesar-code bekomen.

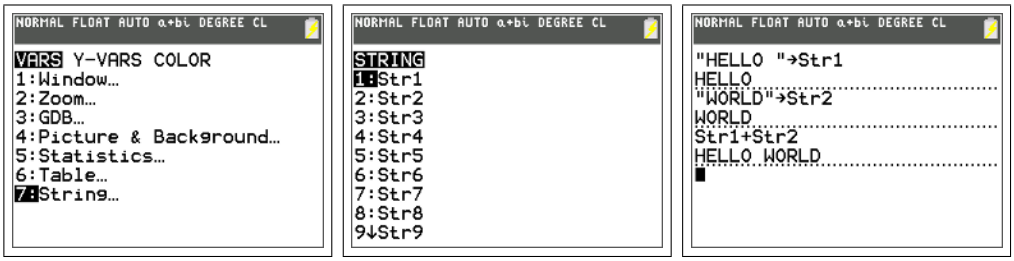
Opdracht 64. Onderstaande tekst werd versleuteld met $K = 17$. Gebruik je *formula* om de oorspronkelijke tekst terug te vinden.

FG VVE UVI VVIJKV CVEKVURXVE
MFVCUV NFCW UV YFEXVI BERXVE,

Opdracht 65. Nog voor Caesar en lang voor Alberti gebruikten de Spartanen reeds een ander hulpmiddel (nml. een *skytale*) om berichten te encoderen en te decoderen. Zoek op wat een *skytale* is en hoe het werkt.

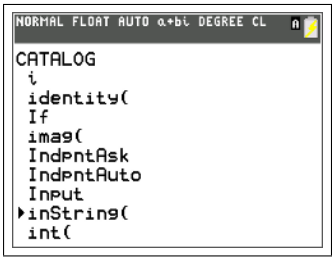
Bij de *skytale* worden de letters van plaats veranderd, het is een **permutatiecode**. Bij de Caesar-code worden de letters echter vervangen door andere, dit is een voorbeeld van een **substitutiecode**.

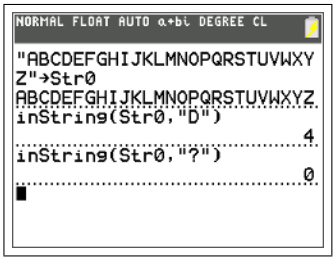
Natuurlijk is het encoderen van een hele teksten soms een lange taak. Tegenwoordig kunnen computers of in dit geval de **TI-84 Plus Color** hulp bieden. Om de Caesar-code te kunnen implementeren is het nodig om met *strings* te leren werken. In de informatica is een string een reeks characters, meestal worden deze tussen aanhalingstekens geplaatst. Op de **TI-84 Plus Color** zijn er tien stringvariabelen die je vindt onder `vars` `[string]`. Strings kan je achter elkaar plaatsen d.m.v. de optelling. Wanneer je strings ingeeft, is het handig om `2nd` `[A-Lock]` te gebruiken zodat je meerdere letters kunt ingeven.



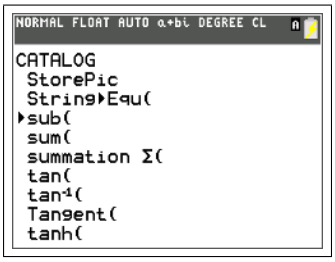
We zullen twee commando's zeer vaak gebruiken. Het eerste is `[inString]`, te vinden via `2nd` `[catalog]` bij de letter I (druk `I` om in de cataloog onmiddellijk naar deze letter te gaan). Deze geeft weer of een bepaalde string

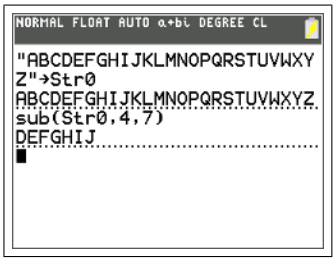
te vinden is in een andere. Indien ja, wordt aangegeven op welke plaats de string teruggevonden kan worden. Indien de tweede string niet voorkomt in de eerste zal het resultaat 0 zijn. We zullen dit gebruiken om de positie van een letter in het alfabet te bepalen. We zullen vanaf nu het alfabet onthouden in `str0`.



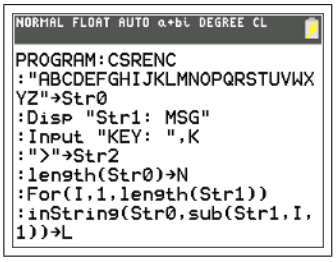


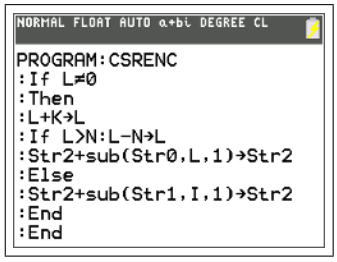
Het tweede commando dat we gaan gebruiken is `[sub]`. Je vindt dit terug via `2nd` `[catalog]` (ga naar de letter T en scroll even terug naar boven). Met dit commando kan je een deel van een string opvragen. In volgend voorbeeld halen we de 7 letters die volgen op D uit ons alfabet.

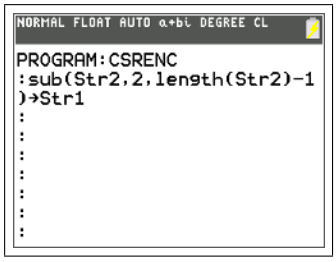




We kunnen nu de **TI-84 Plus Color** gebruiken om de Caesar-code te programmeren. Het programma ziet er als volgt uit.





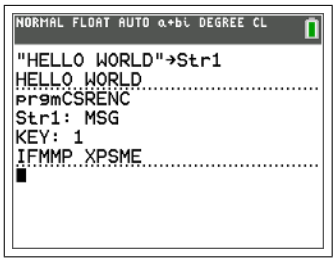


- We beginnen met het te gebruiken alfabet in te geven in `str0`. We gaan ervan uit dat de boodschap in `str1` zit en vragen de gebruiker om de sleutel K.
- Het resultaat komt tijdelijk in `str0`. Omdat de lege string niet bestaat op de **TI-84 Plus Color** beginnen we `str2` met het symbool ">"(te vinden in `2nd` `[test]`). Nadien zullen we dit wegwerken.

- Met `[length]` (cfr. cataloog) bepalen we het aantal letters N in het gebruikte alfabet.
- Hierna begint een `for`-lus.
- Voor elk character in `str1` bepalen we de plaats L in het alfabet.
- Indien het character werd gevonden in `str0` wordt de nieuwe (gecodeerde) positie $L+K$. Indien deze nieuwe positie groter is dan de lengte van het alfabet, wordt er N aftgetrokken. We gebruiken hiervoor de verkorte versie `if...:...` van de `if...then...else...end`-structuur.
- Indien het character niet voorkomt in het alfabet (bijvoorbeeld een punt of een spatie) dan wordt het gewoon overgenomen.
- Tenslotte wordt de gecodeerde boodschap uit `str2` gehaald en in `str1` geplaatst, deze keer zonder het beginteken ">".

Opdracht 66. Maak een stroomdiagram van het programma. Tracht dit zo overzichtelijk mogelijk te doen.

Indien we "HELLO WORLD" encoderen door het alfabet één letter te verschuiven krijgen we:



Opdracht 67. Gebruik het programma om volgende zin te encoderen met sleutel $K = 17$:

WISKUNDE IS ZEER LEUK

Opdracht 68. Om een Caesar-code te decoderen hoef je geen nieuw programma te schrijven. Indien het gebruikte alfabet 26 tekens bevat kan je gewoon de nieuwe sleutel $K' = 26 - K$ gebruiken. Waarom?

Opdracht 69. Volgende bericht werd gecodeerd met het gewone alfabet en $K = 19$. Gebruik de vorige opgave om het te decoderen.

WNL DEHIMX ABC UBC ZKHHMFHX TTG.
WX TKFX ZKHHMFHX LVAKHD SBVA GTTK
PTGM ABC OKTM ATTK HI FXM ANBW XG ATTK.

Opdracht 70. Doordat de spaties niet veranderen kan je zelfs aan het gecodeerde bericht zien hoeveel woorde het oorspronkelijk bericht telt. Pas het programma aan zodat ook de spatie wordt gecodeerd. Versleutel dan het bericht "HELLO WORLD" dmv. $K = 12$ en $K = 21$. Wat merk je?

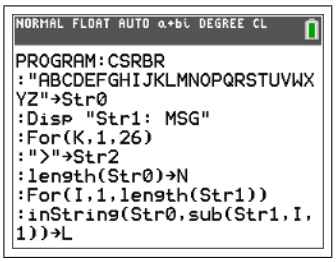
Vanaf nu spreken we af om altijd het gewone alfabet te gebruiken (enkel de letters).

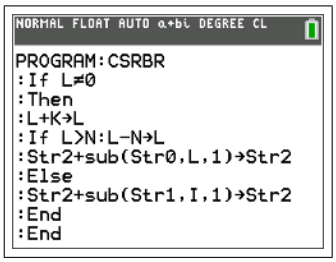
De Caesar-code is een zeer eenvoudige code en is bijgevolg gemakkelijk te kraken. Ten eerste kan men gemakkelijk een statistische analyse van een gecodeerde tekst maken. De meest voorkomende letter in het Nederlands is de "E". De meest voorkomende letter in de gecodeerde tekst stemt dus overeen met een "E". Hieruit vindt men gemakkelijk de sleutel K. Deze methode werkt zeer goed als de tekst lang is.

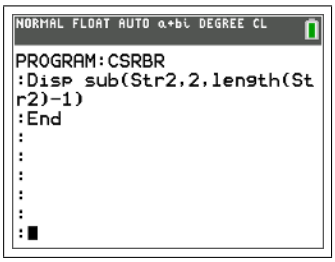
Opdracht 71. Breek volgend gecodeerd bericht.

'KX WL'K PZEOFBI PZEXOJFKHBI JLBQ GB
XIP TLIC JFKPQBKP KLD BBK QLBQGB!'

Opdracht 72. Natuurlijk kan tegenwoordig ook een computer gebruikt worden om de Caesar-code te breken. Ga na hoe volgend programma exact werkt. Maak een stroomdiagram of bespreek regel per regel.







Opdracht 73. Gebruik bovenstaand programma om volgend bericht te kraken.

WTTIPFUOJ PBFR FQ LFZB QFSLX
RFFW 'Y PNSI QFHMYYJ JS YWTP NS JJS BNUOJ
JJS WJATQAJW ZNY MFFW XQNUOJ

Opmerking: Het is niet nodig om alles opnieuw in te geven. Je kan in de programma-editor de recall-functie gebruiken [2nd][rcl] gevolgd door [program][exec] en dan het oorspronkelijk programma. Hierna wordt de volledige code gekopieerd in het nieuwe programma.

3.2 De Vigenère-code

Zoals gezien was de Caesar-code ten tijde van de Romeinen misschien goed, maar met de opkomst van statistiek en de computer biedt deze code geen deftige beveiliging meer. In de 16de eeuw maakte Blaise de Vigère een nieuwe codeermethode populair: een **polyalfabetische substitutiecode**. Hij gebruikte voor deze code een tabel met alle verschuivingen van het alfabet.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

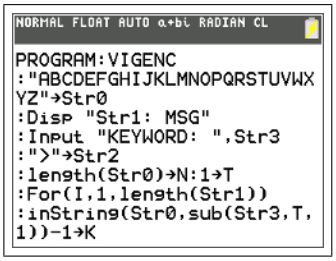
De zender en de ontvanger spreken een sleutelwoord af, bijvoorbeeld CIRKEL. De eerste letter van het bericht wordt gecodeerd volgens het alfabet op lijn C via de Caesar-code. De tweede letter volgens het alfabet op lijn I, enz. ... Wanneer de L werd gebruikt, begin je opnieuw met de C. We

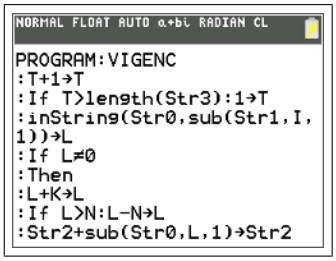
maken hier de afspraak dat een spatie of leesteken ervoor zorgt dat een letter van het sleutelwoord wordt overgeslagen.

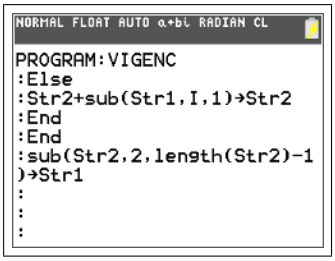
Opdracht 74. Encodeer met de Vigenère-code met seutelwoord CIRKEL de zin "HELLO WORLD".

De Vigenère-code komt dus tot stand door voor elke letter een andere Caesar-code te gebruiken, gebaseerd op de letters van het sleutelwoord. Dus wordt eenzelfde letter (de L in vorig voorbeeld) telkens door andere letters gecodeerd. Hierdoor wordt een statistische analyse onmogelijk. Men is zeer lang overtuigd geweest dat dit een onbreekbare code was vandaar dat de code de bijnaam "*le chiffre indéchiffrable*" kreeg. Pas in de 19de eeuw ontdekte o.a. Charles Babbage (uitvinder van de eerste mechanische computer) een methode om de code te breken.

Omdat de Vigenère-code gebaseerd is op de Caesar-code kunnen we ook deze code programmeren.



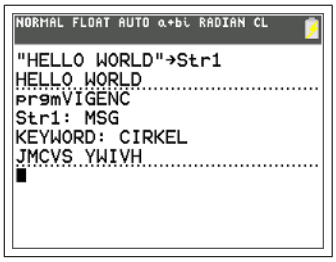




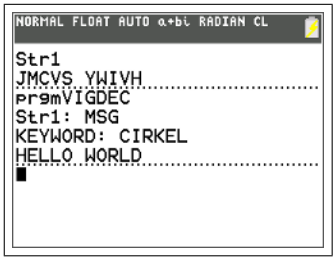
Opdracht 75. Leg regel per regel uit wat dit programma juist doet of maak een stroomdiagram. Enkele tips:

- Het sleutelwoord wordt opgevraagd en in `str3` opgeslagen.
- T houdt bij met welke letter van het sleutelwoord gecodeerd moet worden.
- Na verloop van tijd moet T opnieuw 1 worden als alle letters van het sleutelwoord werden gebruikt.
- De sleutel K van de Caesar-code moet een getal tussen 0 en 25 zijn.

We gebruiken het programma om ons eerste bericht te controleren. Let erop dat je geen aanhalingstekens gebruikt bij het ingeven van het sleutelwoord.



Opdracht 76. Om de Vigenère-code te ontcijferen moet slechts een kleine aanpassing gebeuren aan het programma. Welke? Maak het programma om te ontcijferen (gebruik `[rcl]` om niet alles opnieuw te moeten ingeven).



Opdracht 77. Volgend bericht werd lijn per lijn gecodeerd met het sleutelwoord WISKUNDE. Ontcijfer het.

AV LYAJ, JWXA...LAO VV AZ QYJHIOB!
'PSVFB!' ZAOJ CI DJYFVMO. 'JCR NA OOF
IQBX CUEYPLSAR FKVLTUF ZWV GIYYJDWV!'

Opdracht 78. Als je alle gedecodeerde berichten uit dit hoofdstuk achter elkaar plaatst krijg je een deel van een bekend gedicht. Zoek de auteur en de volledige versie op.

3.3 Foutdetecterende en -verbeterende codes

De bovenstaande codes zijn cryptografische codes. Deze dienen om berichten te beveiligen. Een andere soort codes die voorkomt, zijn de foutdetecterende en -verbeterende codes.

Wanneer een bericht verzonden wordt (bijvoorbeeld een file via het internet), zal het vaak gebeuren dat er onderweg fouten optreden. Bij aankomst is het bericht beschadigd. In deze paragraaf zullen we een eenvoudige manier bespreken om het oorspronkelijke bericht toch te kunnen terugvinden, ondanks mogelijke storingen.

Om te zien hoe krachtig de foutverbeterende codes wel kunnen zijn hoef je maar eens te kijken naar een cd. Deze staat vol muziek die probleemloos

afgespeeld wordt. Als je echter goed naar het cd-oppervlak kijkt, zul je merken dat die vol met krassen staat. Een deel van de informatie is verdwenen, toch kan je cd-speler de fouten verbeteren! Het gaat zelfs zo ver dat je uit een cd een straal kan wegsnijden of met een (dunne) permanente viltstift kan aanduiden, zonder dat de muziek kwaliteit verliest. Probeer maar eens met een oude cd.

Hoe werken zulke foutdetecterende en -verbeterende codes? We geven hier de meest eenvoudige code, de repetitie-code. Veronderstel dat je de letter A wil doorsturen. Indien er storing is, kan het zijn dat je een compleet ander bericht krijgt. Je kan dit schematiseren als volgt.

$$A \longrightarrow \text{ storing } \longrightarrow C$$

Veronderstel nu dat we i.p.v. A het bericht AA sturen, we gebruiken een 2-repetitie-code. We bekomen dus:

$$AA \longrightarrow \text{ storing } \longrightarrow CA$$

Wanneer we het bericht CA binnenkrijgen weten we dat er een fout opgetreden is. De 2-repetitie-code is dus een foutdetecterende code die in staat is om één fout te detecteren. Het nadeel is dan wel dat je oorspronkelijk bericht dubbel zo lang wordt.

Met een 3-repetitie-code kan je een fout niet alleen detecteren maar ook verbeteren.

$$AAA \longrightarrow \text{ storing } \longrightarrow ACA$$

Omdat het bericht ACA een meerderheid van letters A heeft, weet je dat het oorspronkelijk bericht een A was. De 3-repetitie-code is een foutverbeterende code die in staat is om één fout te vebeteren en te detecteren.

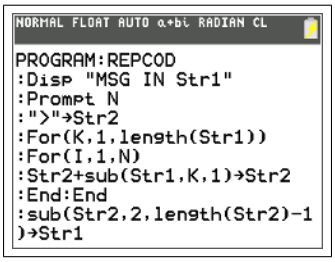
Opdracht 79. Wat weet je als de 3-repetitie-code een resultaat CAZ geeft? Wat kan je concluderen?

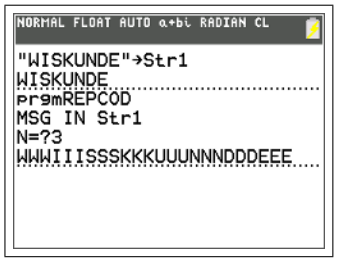
Opdracht 80. Op welke beperking stoot je bij de 3-repetitie-code voor volgend schema?

$$AAA \longrightarrow \text{ storing } \longrightarrow QAQ$$

Opdracht 81. Analyseer de 4-repetitie-code. Geef de nodige voorbeelden.

We zullen nu de N-repetitie-code implementeren op de **TI-84 Plus Color** . Bekijk volgend programma. Elke letter van de string in `Str1` wordt N keer herhaald.

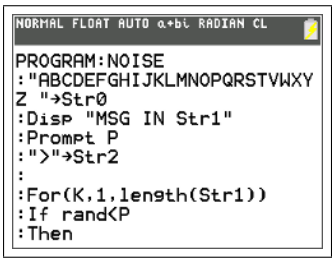


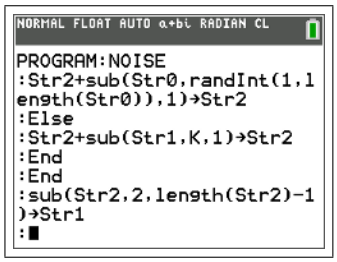


Opdracht 82. Bespreek lijn per lijn bovenstaand programma of maak een stroomdiagram. Geef nog enkele voorbeelden.

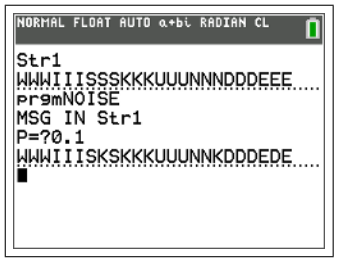
Wanneer een bericht wordt verzonden treden er storingen op. Elk teken in het bericht heeft een bepaalde kans P om verstoord te worden.

Opdracht 83. Met `math` `[prb]` `[rand]` kan de **TI-84 Plus Color** een willekeurig getal in $[0, 1[$ genereren. Met `math` `[prb]` `[randInt]` kan een willekeurig geheel getal gekozen worden en dus ook een willekeurige letter uit het gebruikte alfabet (hier met de spatie). Hiermee kan het programma `NOISE` (Engels voor storing) de verstoring van een transmissie nabootsen. Bespreek elke regel van dit programma of maak een stroomdiagram.





Opdracht 84. We simuleren nu een storing van 10% op de transmissie.



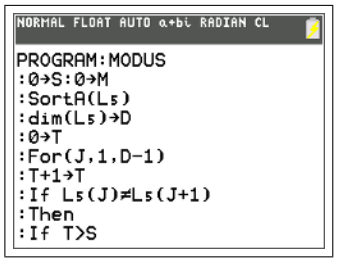
Ga zelf eens na tot welk percentage storing je kan gaan alvorens de boodschap verloren gaat.

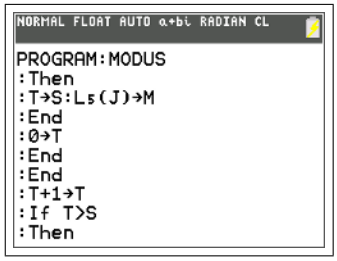
We beschouwen de N-repetitie-code ($N = 7$):

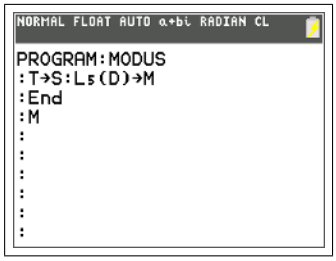
$$AAAAAAA \longrightarrow \text{ storing } \longrightarrow AABZABA$$

Om deze te gebruiken om een bericht terug te vinden moeten we het bericht $AABZABA$ vertalen naar A. Hiervoor moeten we in het bekomen bericht de meest voorkomende letter terugvinden. We zullen eerst het bericht vertalen naar getallen $1, 1, 2, 26, 1, 2, 1$ door de positie van elke letter in het alfabet te gebruiken. In de gegevens $1, 1, 2, 26, 1, 2, 1$ moeten we nu het meest voorkomende getal terugvinden. Uit de statistiek weten we dat dit de modus is.

Opdracht 85. Ga na hoe onderstaand programma uit de lijst L_5 de modus M bepaalt. Leg elke regel uit of maak een stroomdiagram.

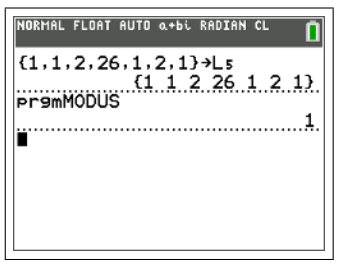




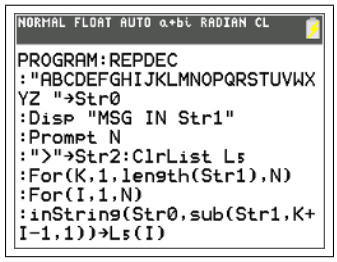


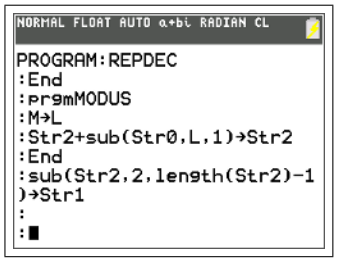
Enkele tips:

- De lijst wordt gesorteerd.
- Er wordt via T geteld hoeveel keer een getal voorkomt.
- Indien dit meer voorkomt dan het vorige meest voorkomende getal M (dat S keer voorkwam), dan worden S en M aangepast.
- Buiten de `for`-lus gebeurt een uiteindelijke controle voor het laatste voorkomend getal.
- Uiteindelijk staat in M het meest voorkomend getal uit de lijst.



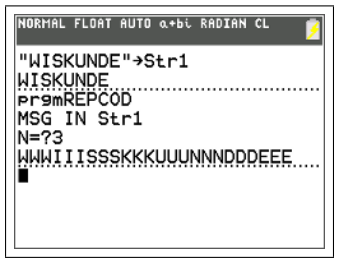
We zijn nu in staat om een programma te schrijven dat een N-repetitiecode decodeert.

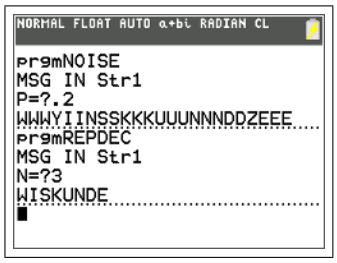




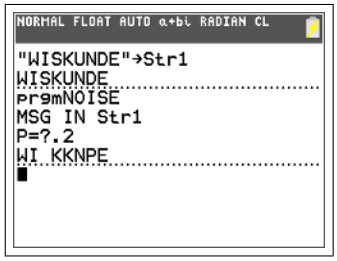
Opdracht 86. Bespreek lijn per lijn wat dit programma doet of maak een stroomdiagram.

Opdracht 87. Bekijk volgend voorbeeld en maak er zelf ook een aantal.





Zonder de foutverbeterende code en met dezelfde storingsgraad zou dit kunnen gebeuren:



Hoofdstuk 4

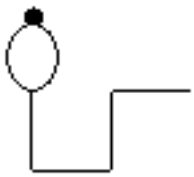
Algoritmische planten

In de biologie bestudeert men reeds eeuwen planten. Dankzij de moderne technologie en een beetje wiskunde kan men verschillende plantenfamilies omvormen tot formules. Hiermee kan men op computer en in de cinema nieuwe levensechte planten op het scherm toveren. We geven hier voorbeelden die op de **TI-84 Plus Color** gemaakt kunnen worden. Hiervoor zullen we een eigen programmeertaal maken.

4.1 Turtle graphics: een eigen programmeertaal maken!

Vooraleer we zelf planten gaan maken zullen we eerst zelf een kleine programmeertaal ontwikkelen om grafieken mee te maken.

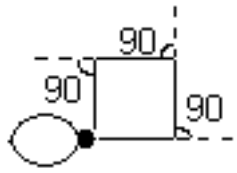
We beschouwen een schildpad in een veld. Het diertje verplaatst zich vooruit over een welbepaalde lengte. Tussen twee verplaatsingen kan ze van richting veranderen, ze kan zich draaien over een bepaalde hoek. Op die manier laat ze een spoor achter.



Je kan de schildpad nu "programmeren" om een bepaalde tekening te maken door te zeggen wanneer ze vooruit moet en wanneer ze moet draaien en over

welke hoek. We spreken af dat 0 wil zeggen dat ze 1 m vooruit moet gaan en dat 90 wil zeggen dat ze over 90 graden moet draaien. We kunnen nu een vierkant tekenen door de schildpad volgende opdracht te geven.

$$0, 90, 0, 90, 0, 90, 0$$



De afspraken die we hebben gemaakt leggen een eenvoudige programmeertaal vast, het programma om een vierkant te tekenen bestaat uit de bovenstaande opdrachten.

Opdracht 88. Wat is het programma om een gelijkzijdige driehoek te maken? En een rechthoek met lengte 3 m en breedte 2 m?

We gaan onze programmeertaal uitbreiden met twee nieuwe commando's. Indien in het programma het getal 1 staat, wordt de huidige positie en richting van de schidpad onthouden. Indien -1 tegengekomen wordt, dan keert de schildpad naar het laatst onthouden punt terug. Merk op dat het hierdoor niet meer mogelijk is om over een hoek van slechts $1°$ te draaien.

Opdracht 89. Wat doet volgend programma?

$$0, 90, 0, 1, 90, 0, 90, 0, -1, -90, 0, 90, 0, 90, 0, 90, 0$$

We zullen deze programmeertaal implementeren op de **TI-84 Plus** . We gaan ervan uit dat de schildpad altijd in $(0, 0)$ start, met haar neus in de positieve x-richting. De stap die we telkens zetten, zal in de veranderlijke F staan. Het programma is een lijst getallen, we zullen hiervoor de lijst L_6 gebruiken, zodat de andere lijsten vrij blijven. De richting waar de schildpad naartoe gaat wordt bijgehouden in de variabele T (de hoek gemeten vanaf de positieve x-as). De positie van de schildpad wordt bijgehouden in (A, B). Het programma ziet er als volgt uit.

```
NORMAL FLOAT AUTO REAL DEGREE MP
PROGRAM:TURTLE
:ClrDraw:AxesOff:Degree
:ClrList L3,L4,L5
:0→A:0→B:0→T:0→M
:For(I,1,dim(L6))
:If L6(I)=0
:Then
:A→C:B→D
:A+Fcos(T)→A
:B+Fsin(T)→B
```

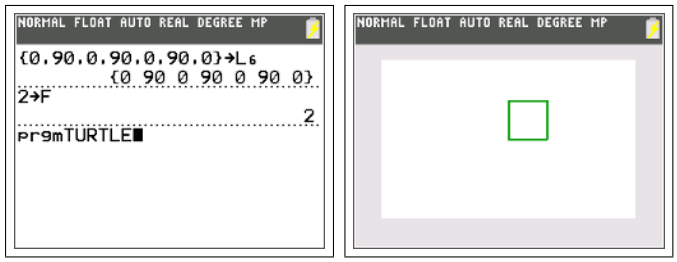
```
NORMAL FLOAT AUTO REAL DEGREE MP
PROGRAM:TURTLE
:Line(C,D,A,B,GREEN)
:End
:If L6(I)=1
:Then
:M+1→M
:A→L3(M):B→L4(M):T→L5(M)
:End
:If L6(I)=-1
:Then
```

```
NORMAL FLOAT AUTO REAL DEGREE MP
PROGRAM:TURTLE
:L3(M)→A:L4(M)→B:L5(M)→T
:M-1→M
:End
:If abs(L6(I))>1
:Then
:T+L6(I)→T
:End
:End
:
```

- Nadat we het scherm hebben leeggemaakt, schakelen we de **TI-84 Plus** om naar graden.
- De startpositie is de oorsprong en de schildpad is naar de positieve x-as gericht. Het geheugen M van de schildpad staat initieel op 0.
- Nu wordt het schildpad-programma uit lijst L_6 gelezen.
- Als in L_6 een nul staat, wordt de nieuwe positie berekend d.m.v. de stap F en de hoek T. Er wordt dan een lijnstukje getekend naar deze nieuwe positie.
- Indien 1 wordt gelezen uit L_6 zal de huidige positie en hoek in het geheugen geplaatst worden via L_3, L_4 en L_5.
- Indien -1 wordt tegengekomen zal een nieuwe positie uit het geheugen gehaald worden en is dit vanaf nu de huidige positie.
- Als L_6 een andere waarde heeft, is dit een hoek en wordt de nieuwe richting in T opgeslagen.

Opdracht 90. Maak een stroomdiagram van het programma. Tracht dit zo overzichtelijk mogelijk te doen.

Als we bovenstaand schildpad-programma willen uitvoeren om een vierkant te tekenen, steken we het in L_6 en leggen de afstand F vast. Daarna roepen we onze programmeertaal op. Je moet natuurlijk wel rekening houden met de grenzen van het grafisch venster.



Opdracht 91. Gebruik je schildpad-programma om een gelijkzijdige driehoek en een rechthoek te tekenen.

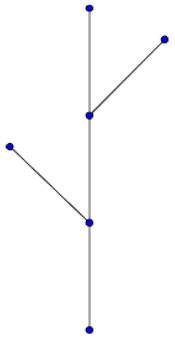
Opdracht 92. Met welk schildpad-programma kan je een rechthoekige driehoek tekenen? Welke stelling(en) heb je hiervoor nodig?

Opdracht 93. Schrijf een schildpad-programma om een mannetje te tekenen.

4.2 Lindenmayer-systemen

Aristid Lindenmayer (1925-1989) was een Hongaars bioloog die werkt aan de universiteit van Utrecht. Hij bestudeerde de groeistructuur van planten. Hij ontwikkelde volgende methode om de vertakkingen van een plant te modeleren. Het onderstaand plantje stelde hij voor door volgende rij symbolen:

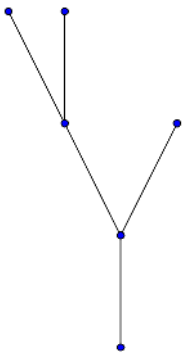
$$L[+L]L[-L]L$$



De betekenis van deze symbolen legde hij vast als volgt.

- L geeft een takje aan als een recht lijnstuk in een bepaalde richting
- [geeft het begin van een zijtakje aan
- $+$ geeft een richtingsverandering weer naar links
- $-$ geeft een richtingsverandering weer naar recht
-] geeft het einde van een zijtakje aan

Opdracht 94. Met welke rij symbolen komt volgende plant overeen?



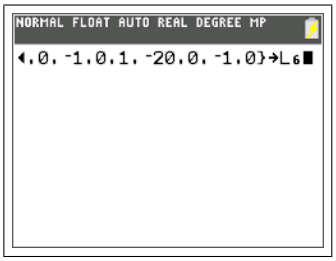
Het is tamelijk duidelijk dat een Lindenmayer-beschrijving van een plant omgezet kan worden naar een `Turtle`-programma:

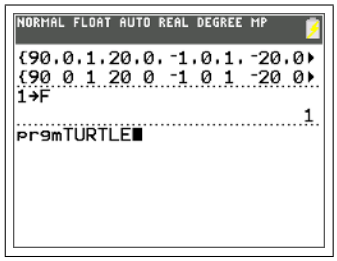
- L wordt vertaald naar 0, d.w.z. een stap vooruit zetten
- [en] komen overeen met 1 en -1, het onthouden en terugkeren naar een voorgaande plaats
- $+$ en $-$ houden draaiingen in, we kunnen bijvoorbeeld kiezen voor $+20°$ en $-20°$.

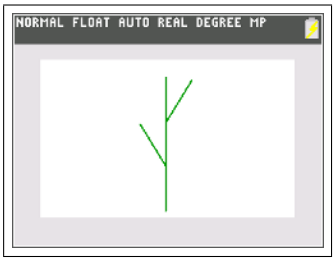
We merken wel op dat planten in het algemeen verticaal groeien, we moeten ons programma dus starten met de richting van de schildpad te veranderen $(90, \ldots)$. Indien we het eerste takje $(L[+L]L[-L]L)$ van dit hoofdstuk vertalen naar een programma bekomen we

$$90, 0, 1, 20, 0, -1, 0, 1, -20, 0, -1, 0$$

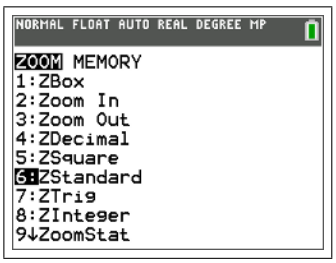
We proberen dit uit.

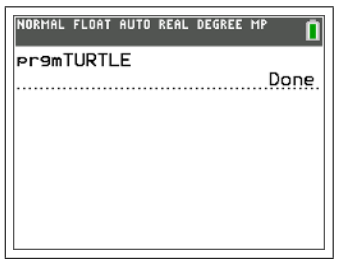


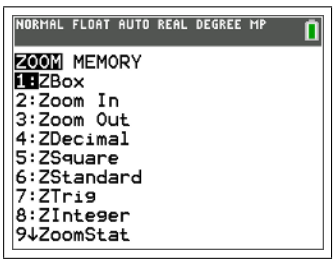


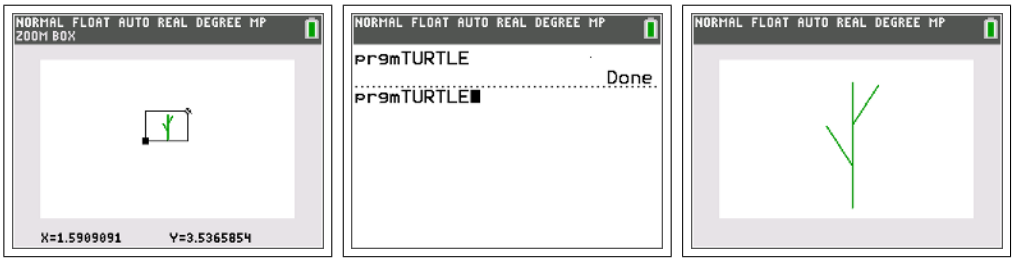


Merk op dat het vaak niet evident is om een correct venster te vinden zodat de grafiek mooi over heel het scherm. Om dit te doen kan je volgende stappen gebruiken. Maak eerst de grafiek met `zoom` `[zdecimal]` of `zoom` `[zstandard]`. Gebruik daarna `zoom` `[zbox]` om de grafiek mooi te centreren. Roep tenslotte het programma opnieuw aan.

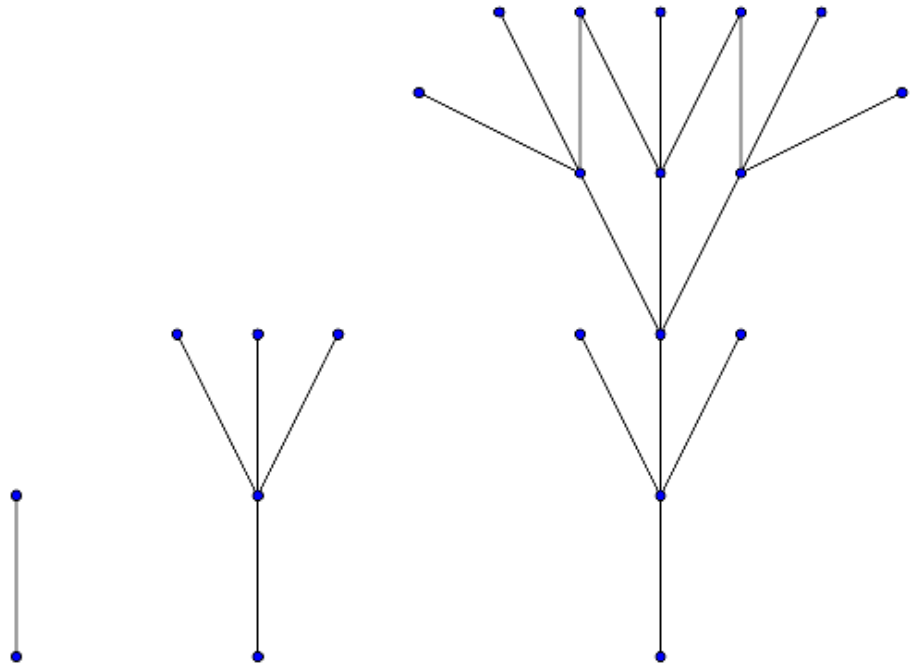








De grote ontdekking van Lindenmayer was dat plantengroei werd beschreven door eenvoudige substitutieregels. Een plantje begint te groeien aan de hand van één scheut. Deze zal dan vertakken in een bepaald patroon. Tijdens de verdere groei zal elke tak opnieuw volgens datzelfde patroon vertakken.



De eerste stap bestaat uit één enkele tak.

$$L$$

De tweede groeistap geeft het groeipatroon. Elk uiteinde van een tak splitst in drie en de plant groeit verder vanuit de middenste tak.

$$L[+L][-L]L$$

Wanneer de plant blijft verdergroeien zullen onderaan kleine nieuwe takjes groeien, terwijl de bestaande takken verder in drie splitsen. De plant wordt nu beschreven door

$$L[+L][-L][+L[+L][-L]L][-L[+L][-L]]L[+L][-L]L$$

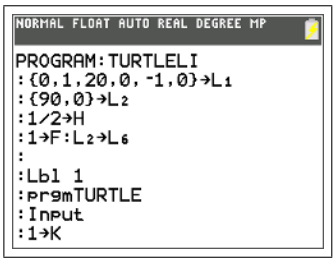
Lindemayer merkte op dat dit ingewikkeld patroon tot stand kwam door in de vorige stap telkens L te vervangen door het patroon $L[+L][-L]L$. De plant wordt dus volledig beschreven door

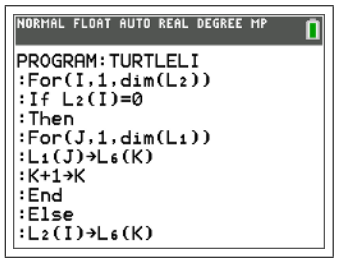
- de startfase: L
- het groeipatroon: $L \to L[+L][-L]L$

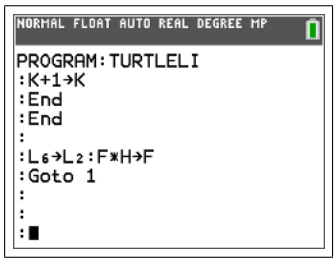
Opdracht 95. Teken de plant met startfase L en groeipatroon

$$L \to L[+L] - L$$

Deze substitutiemethode om planten te tekenen kunnen we nu programmeren in de **TI-84 Plus Color** .

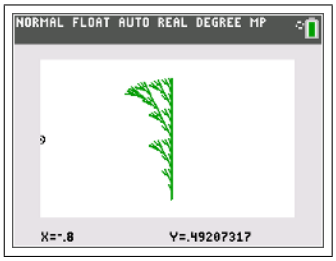






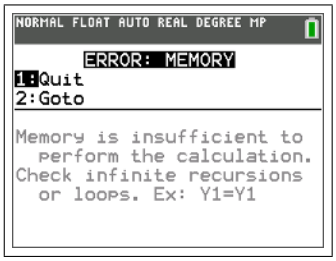
- Eerst wordt het patroon opgeslagen in L_1, de startfase komt in L_2.
- De schaalfactor H is in dit geval $\frac{1}{2}$, omdat de plant in stappen groeit zullen we deze schaalfactor gebruiken om ervoor te zorgen dat de tekening niet buiten het scherm komt. Afhankelijk van de plant moet deze factor aangepast worden. Indien de plant na één stap tweemaal zo groot wordt is de factor $\frac{1}{2}$.
- De staplengte F is 1. Het beginprogramma voor de schildpad is de startfase en komt in L_6.
- Daarna wordt een lus gestart.
- `prgmTURTLE` wordt aangeroepen om de eerste tekening te maken, met `prgm` `[io]` `[input]` wordt gewacht tot de gebruiker op `enter` drukt voor de berekening van de volgende stap.

- Hierna volgt een `for`-lus, voor elk element uit L_2 wordt nagegaan of dit een lijnstukje is (0).
- Als het een lijnstukje is, wordt d.m.v. een andere `for`-lus het patroon uit L_1 naar het nieuwe programma L_6 gekopiëerd.
- Anders wordt het commando gewoon overgenomen.
- Met K hou je de lengte van het nieuwe programma bij.
- Uiteindelijk staat in L_6 het programma voor de volgende stap.
- Omdat dit het begin is van een volgende herhaling maken we een kopie ervan in L_2, daarna wordt de nieuwe staplengte gegeven aan de hand van de schaalfactor en wordt de lus gesloten.
- De nieuwe tekening zal gemaakt worden en de volgende zal worden berekend.
- Het programma zal doorlopen tot je het met [on] onderbreekt of tot het geheugen vol is (lijsten kunnen maar 999 elementen bevatten). Het resultaat na 5 stappen is volgende plant.

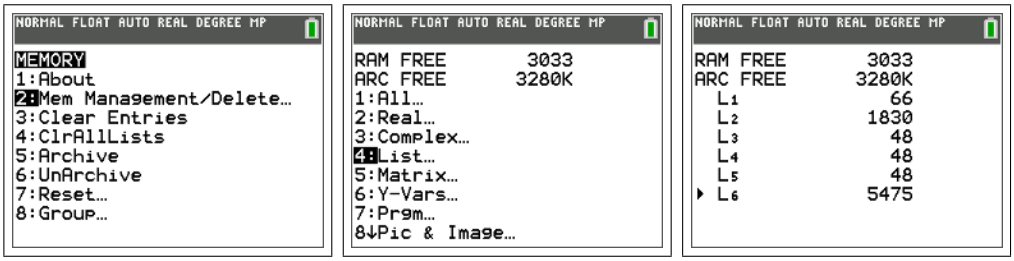


Opdracht 96. Maak een stroomdiagram van het programma. Tracht dit zo overzichtelijk mogelijk te doen.

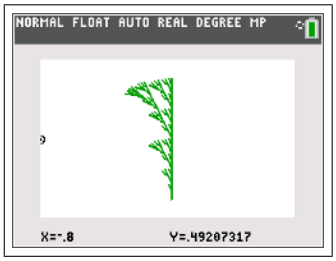
Merk op dat 5 stappen al relatief veel zijn. Meestal zal reeds na drie of vier stappen gestopt moeten worden. Indien je in bovenstaand geval een zesde stap wenst uit te rekenen stoot men op de grenzen van de **TI-84 Plus Color** .



Het geheugen van het toestel is nu zo vol dat er problemen zijn om verder te werken. Om het geheugen te ledigen gebruik je best [2nd][mem][Mem Management/Delete][lists] en selecteer je [L_6] (de meest volle lijst). Druk dan [Del] om ze weg te vegen.



Opdracht 97. We beschouwen de net berekende plant.

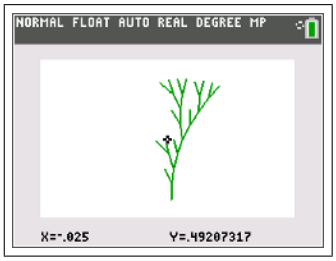


Wat is hier het groeipatroon? Geeft dit een natuurlijk effect? Waarom/Waarom niet?

Opdracht 98. Wat gebeurt er indien men $\{80, 0\} \to L_2$ gebruikt als startfase?

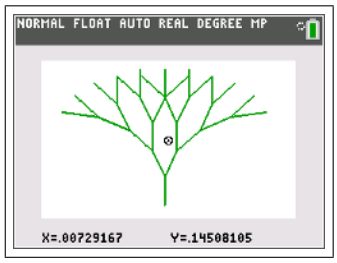
Opdracht 99. Wat gebeurt er indien men $\{1, 110, 0, -1, 80, 0\} \to L_2$ gebruikt als startfase? Wat heeft dit als gevolg voor het geheugen van het rekentoestel?

Opdracht 100. Beschouw nu de plant met startfase $\{90, 0\} \to L_2$ en met het iets natuurlijkere groeipatroon $\{0, 1, 20, 0, -1, -5, 0\} \to L_1$.

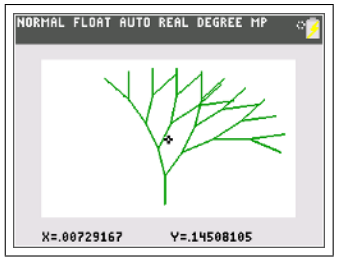


Tracht dit om te zetten gebruik makend van de Lindenmayer-code. Wat is het probleem. Waarom is ons programma iets uitgebreider dan het Lindenmayersysteem? Hoe kan je dit systeem aanpassen?

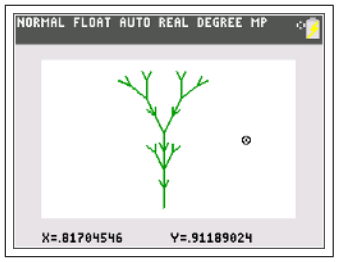
Opdracht 101. Tracht volgende boom na te maken.



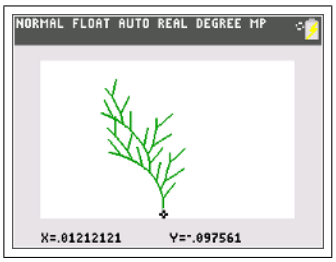
Opdracht 102. Wat moet je in de vorige opdracht veranderen om deze boom te krijgen?

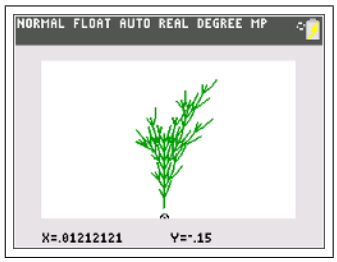


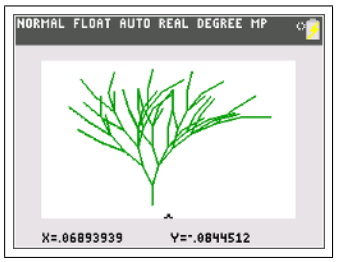
Opdracht 103. Wat moet je in opdracht 101 veranderen om deze boom te krijgen?



Opdracht 104. Maak nu zelf enkele andere planten. Enkele mogelijke inspiraties:







Tenslotte is het een leuk weetje dat de algoritmen van Lindenmayer niet alleen in de biologie haar toepassingen heeft gevonden maar ze worden vooral gebruikt in de filmindustrie! Zo zijn films als **Avatar** of **Star Wars** bekend om weelderige beelden van wilde oerwouden op verre planeten. Elke boom van zo'n oerwoud werd berekend volgens de technieken van Lindenmayer. Een deel van het onderzoek omtrent de wiskundige modelisatie van planten werd dan ook gevoerd door mensen van Lucasfilm. Niet enkel wetenschappers hebben de wiskunde nodig!

Bijlage A

Eventjes leren programmeren

Je kan de capaciteiten van de **TI-84 Plus Color** uitbreiden door zelf een programma toe te voegen. In tegenstelling tot hetgeen vaak wordt gezegd, is programmeren voor de **TI-84 Plus Color** niet echt moeilijk. Als leerlingen zelf eens een programma behandelen tijdens de lessen, krijgen ze inzicht in hoe een grafisch rekenmachine werkt. Bovendien is vandaag de dag de belangrijkste toepassing van de wiskunde de informatica, in ruime zin. Van grafische rekenmachine tot statistische computerprogramma's, via gsm's, mp3-spelers, gameboy's en playstations, computer games en internettoepassingen, in bijna elke moderne technologie zit meer dan 50% wiskunde. Zonder wiskunde geen moderne technologie!

De programmeertaal die in de **TI-84 Plus Color** zit, is TIBasic. Dit is een dialect van BASIC, een programmeertaal waarmee Bill Gates (Microsoft) zijn faam heeft verworven. BASIC (Beginners All-purpose Symbolic Instruction Code) werd ontworpen om ook de leek in staat te stellen om kleine programma's te schrijven. Het TIBasic-dialect is trouw aan deze filosofie: er is geen enkele programmeer-ervaring nodig om programma's te schrijven voor de **TI-84 Plus Color** . Enkel een beetje doorzettingsvermogen en zelfvertrouwen is nodig.

Om een programma te schrijven ga je naar `prgm`. Je kunt hier kiezen om een programma uit te voeren (`[exec]`), te veranderen (`[edit]`) of om een nieuw programma te schrijven (`[new]`). Indien je de laatste keuze maakt wordt er naar een naam gevraagd. Nadien kom je op de editor uit, waar je jouw programma kan invoeren. De commando's die met het programmeren te maken hebben, zitten nu onder `prgm`. We geven hier een kort overzicht van de nuttigste programmeerfuncties.

[ctl]	
[if]	Eerste vorm: `:if` *voorwaarde* `:` *commando*
[then]	Tweede vorm: uitgebreide `if` structuur:

[else]	`:if` *voorwaarde*
	`:then`
	`:` *commando's*
	`:else`
	`:` *commando's*
	`:end`

[for]	om lussen te maken
	`:for(` *var, beginwaarde, eindwaarde [, stapgrootte]* `)`
	`:` *commando's*
	`:end`

[end]	om bovenstaande blokken te eindigen

[i/o]	
[disp]	om een waarde/string op het scherm te printen
[prompt]	om de waarde van een variabele te vragen aan de gebruiker
[input]	om een tekst op het scherm te tonen en een waarde te vragen aan de gebruiker, deze kan ook een functie zijn. `:input` *"text", variabele* (bijv. y_1)

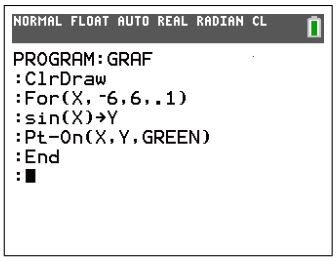
Nu we de nodige commando's kennen, kunnen we een zeer eenvoudig voorbeeld behandelen. Dit voorbeeld geeft inzicht over hoe de **TI-84 Plus** grafieken maakt.

Opdracht 105. Gebruik een `for`-lus om een programma te schrijven dat de grafiek van een functie (bijvoorbeeld de sinusfunctie) maakt d.m.v. het volgende algoritme.

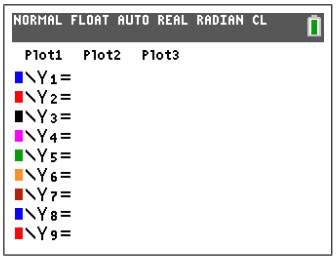
```
voor x gaande van -6 tot 6 met een stap van 0.1
bereken y=sin(x)
teken het punt (x,y)
sluit de lus
```

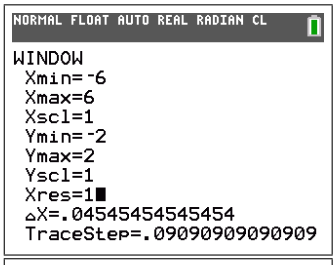
Gebruik de grafische commando's `2nd` `[draw]` `[clrdraw]` (om een leeg scherm te krijgen) en `2nd` `[draw]` `[point]` `[pt-on]` om een punt te tekenen. Vergeet niet van via `y=` alle functies weg te halen en om met `window` de grenzen aan te passen!

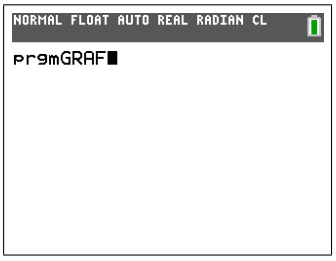
Het programma telt exact vijf regels:

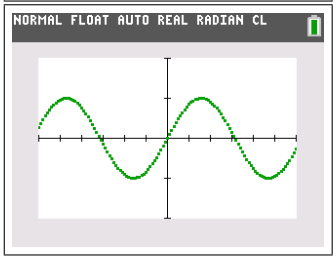


Het resultaat is hetzelfde als hetgeen we zouden verkrijgen door de ingebouwde functies te gebruiken.





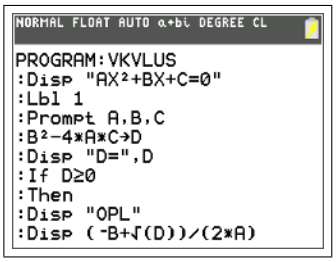


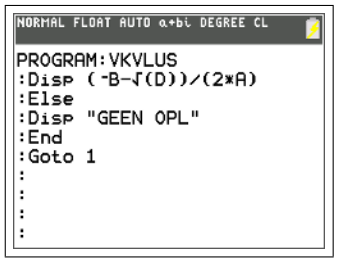


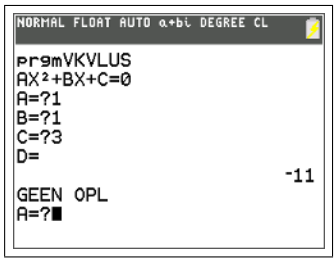
Bijlage B

Bespreken van een programma

Tijdens het maken van de opdrachten wordt vaak gevraagd om een programma te bespreken. Dit kan je best op volgende manieren doen. Beschouw volgend programma dat herhaaldelijk een tweedegraadsvergelijking oplost.







Om dit programma te bespreken kan je twee methoden gebruiken.

Regel per regel

1: De opgave wordt afgedrukt zodat de gebruiker de betekenis van de parameters begrijpt.

2: Een `label` wordt gebruikt om een lus te starten.

3: Aan de hand van `prompt` wordt de gebruiker gevraagd om waarden voor A, B en C in te geven.

4-5: De discriminant wordt berekend en afgedrukt.

6: Een `if`-structuur wordt gestart.

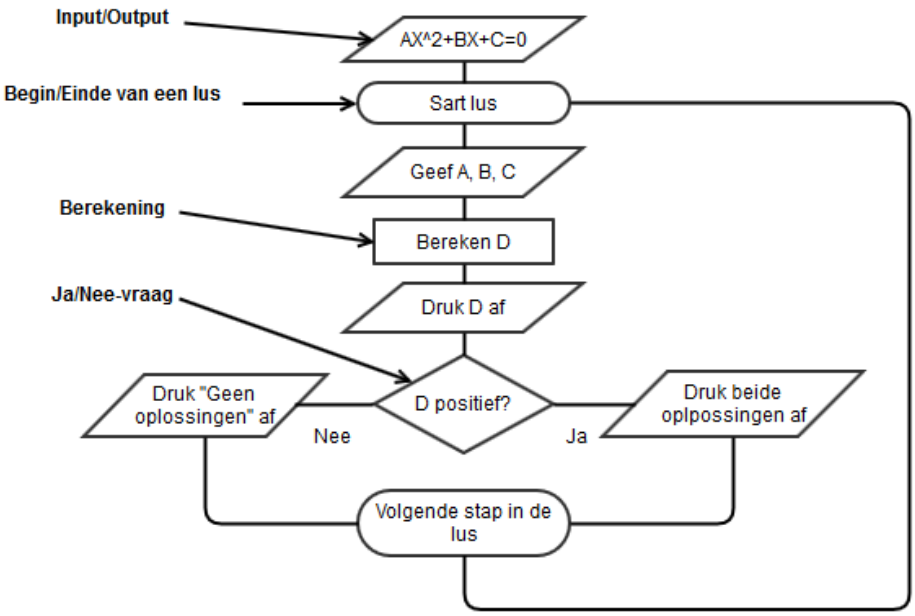
7-10: Indien $D \geq 0$ worden de twee oplossingen bepaald en afgedrukt.

11-13: Indien $D < 0$ zijn er geen oplossingen en wordt dit afgedrukt.

14: De lus wordt gesloten en het programma keert terug naar de tweede regel.

Stroomdiagram

Een andere methode hiertoe is een **stroomdiagram** of **flowchart**. Hierbij worden geometrische vormen gebruikt om bepaalde programmablokken voor te stellen (input/output, lussen, berekeningen, jan/nee-vragen).



Bijlage C

Oplossingen van de opdrachten

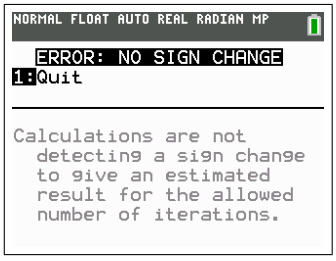
4: Voor de functie $f(x) = ax^3 + bx^2 + cx + d$ is de cubische discriminiant

$$\Delta = 18abcd - 4b^3d + b^2c^2 - 4ac^3 - 27a^2d^2$$

- Indien $\Delta > 0$ dan zijn er drie verschillende nulwaarden in $\mathbb{R}$.
- Indien $\Delta = 0$ dan zijn er meervoudige nulwaarden in $\mathbb{R}$.
- Indien $\Delta < 0$ dan is er slechts één nulwaarde in $\mathbb{R}$.

7,8: Zij f een veeltermfunctie van oneven graad. Voor een voldoende groot getal c hebben $f(-c)$ en $f(c)$ een verschillend teken. Er bestaat dus een nulwaarde.

9: Wanneer je een dubbele nulwaarde tracht op te sporen met de **TI-84 Plus Color** krijg je volgende foutmelding.



De foutmelding `no sign change` wijst erop dat de **TI-84 Plus Color** intern een algoritme gebruikt dat gebaseerd is op de stelling van Bolzano en dus een tekenverandering vereist.

12: Gebruik het programma om de nulwaarden te vinden van $y = x^2 - 2$ en $y = x^2 - x - 1$.

14:

```
NORMAL FLOAT AUTO REAL RADIAN MP
PROGRAM:REGFALSI
:Disp "FFTIE Y1"
:Prompt A,B,N
:
:For(K,1,N)
:(A*Y1(B)-B*Y1(A))/(Y1(B)-
Y1(A))→C
:
:If Y1(A)*Y1(C)<0
:Then
```

```
NORMAL FLOAT AUTO REAL RADIAN MP
PROGRAM:REGFALSI
:C→B
:Else
:C→A
:End
:End
:Disp "NULPT LIGT TSSN",A,
"EN",B
:
:
```

15: In de meeste gevallen zal voor eenzelfde aantal iteraties en eenzelfde startinterval de regula falsi een betere benadering geven dan de bisectiemethode

19:

```
NORMAL FLOAT AUTO REAL RADIAN MP
PROGRAM:NEWTRAPH
:Disp "FTIE Y1"
:Prompt Z,N
:For(K,1,N)
:Z-Y1(Z)/nDeriv(Y1,X,Z)→Z
:End
:Disp "NULW",Z
:
```

23: Passen we de methode van Newton-Raphson toe op $f(x) = x^2 - n$, dan krijgen we

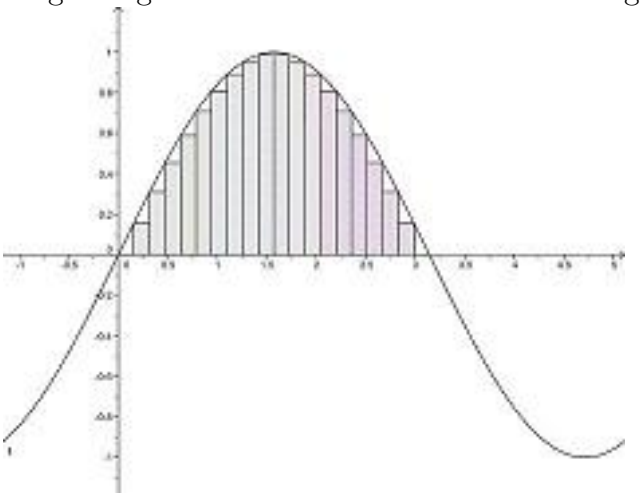
$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} = x_n - \frac{x_n^2}{2x_n} = x_n - \frac{x_n}{2} + \frac{a}{2x_n} = \frac{x_n}{2} + \frac{a}{2x_n}$$

We bekomen dus

$$x_{n+1} = \frac{1}{2}(x_n + \frac{a}{x_n})$$

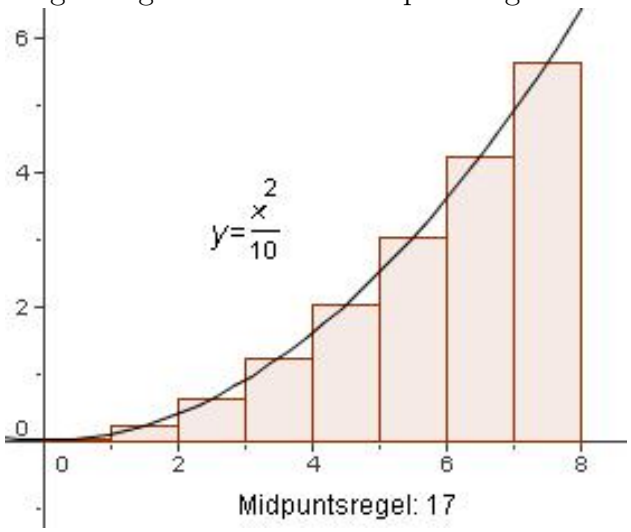
Dit is juist wat het algoritme doet.

24: Volgende grafiek stelt de ondersom-benadering voor:

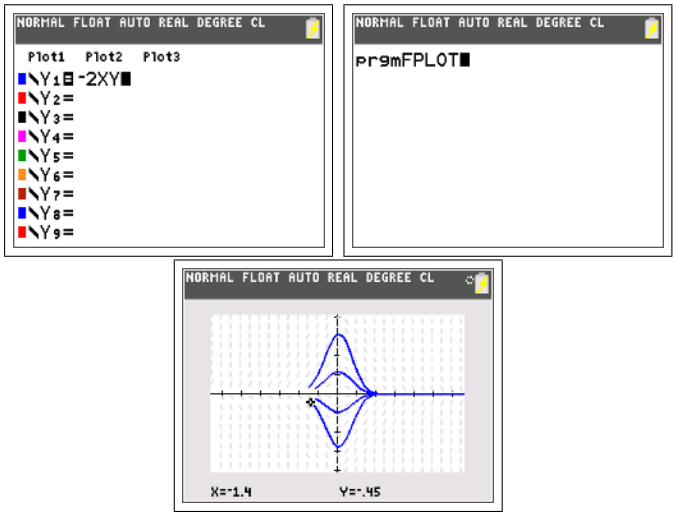


26: Voor grote waarden van n is het interval $[x_{i-1}, x_i]$ zeer klein, zodat mag worden aangenomen dat op dit interval f strikt stijgend of dalend is. Het minimum van f op dit interval bevindt zich dan in één van de eindpunten: $\min\{f(x_{i-1}), f(x_i)\}$.

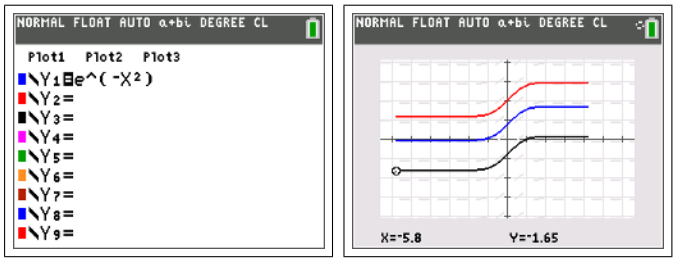
29: Volgende grafiek stelt de midpuntsregel-benadering voor:



36: Met de **TI-84 Plus Color** geeft de eerste vergelijking volgend resultaat.

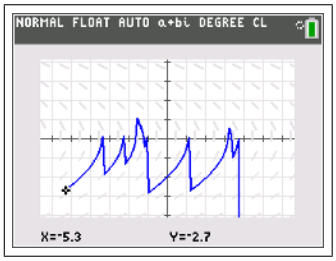


Door de veranderlijken af te zonderen en dan te integreren bekomt men de algemene oplossing $y = ce^{-x^2}$. De tweede vergelijking wordt met de **TI-84 Plus Color** als volgt opgelost.



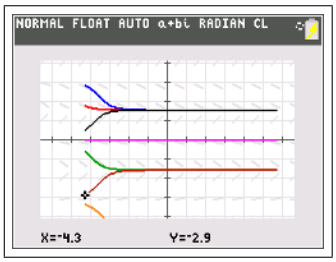
Deze differentiaalvergelijking kan men, hoe eenvoudig ze ook is, onmogelijk oplossen. De oplossing wordt immers gegeven door $y = \int e^{-x^2}\,\mathrm{d}x$ en deze integraal kan niet uitgerekend worden aan de hand van de elementaire functies. In zulk geval (en dit komt veelvuldig voor - het is de normale verdeling uit de kansrekening) is men dus verplicht numerieke integratie toe te passen.

37: Het resultaat is het volgende.



Een korte berekening toont dat de oplossing van deze vergelijking $y = \sqrt{-4x + c}$ is. Dit is een parabool, met as $y = 0$. In haar top is de raaklijn dus verticaal en heeft de differentiaalvergelijking geen zin. Ons algoritme, dat met benaderde waarde werkt, zal in deze punten uiterst gevoelig zijn voor de kleinste fout en zal niet meer van toepassing zijn.

38: Onze programma's geven volgend resultaat.



Men ziet dat op lange termijn, afhankelijk van de beginvoorwaarde er een evenwicht zal zijn in $y = \pm\frac{\pi}{2}$. Tenzij de beginvoorwaarde op de x-as ligt, dan zal de triviale oplossing $y = 0$ gevonden worden. Het evenwichtspunt 0 is afstotend, de anderen zijn aantrekkend.

39: Een veelterm van graad n in x is een uitdrukking van de vorm

$$a_n x^n \ldots a_1 x + a_0$$

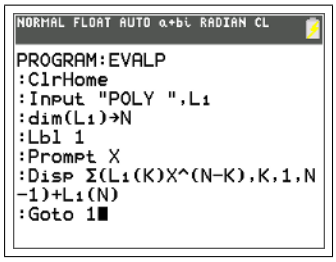
waarbij de coëfficiënten a_i getallen zijn en $a_n \neq 0$. $\mathbb{R}[x]$ is de verzameling van alle veeltermen in x met coëfficiënten in $\mathbb{R}$. $\mathbb{Q}[x]$ is de verzameling van alle veeltermen in x met coëfficiënten in $\mathbb{Q}$.

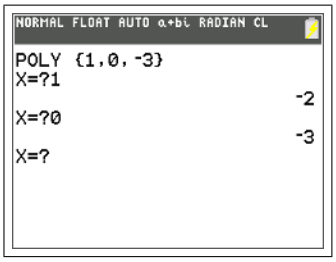
40: Als twee veeltermen een verschillende graad hebben, dan zijn de overeenstemmende lijsten niet even lang. Bijgevolg zal je een foutmelding krijgen als je de lijsten wilt optellen.

42: Dit is de vermenigvuldiging met x en met x^2.

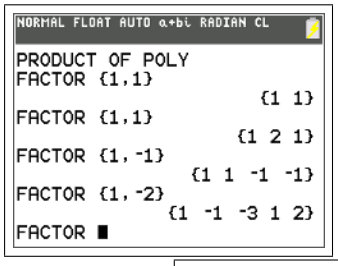
44: In dit geval komt men de macht 0^0 tegen, deze is niet gedefinieerd!

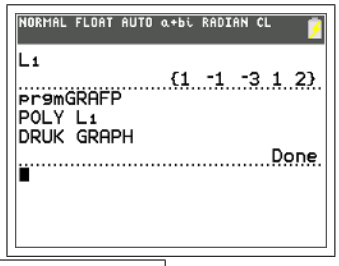
45: We lossen het probleem als volgt op.

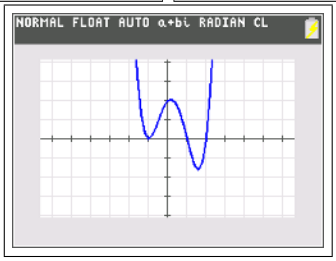




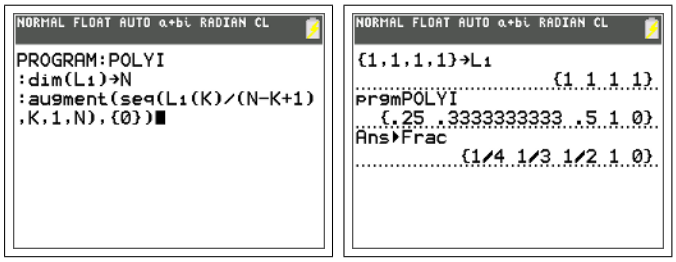
50: Bijvoorbeeld:





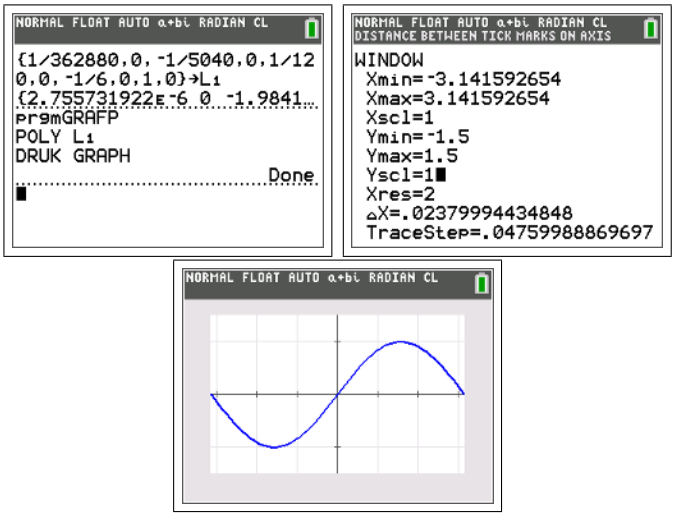


58: Volgend programma bepaalt een primitieve van de veelterm in L_1.



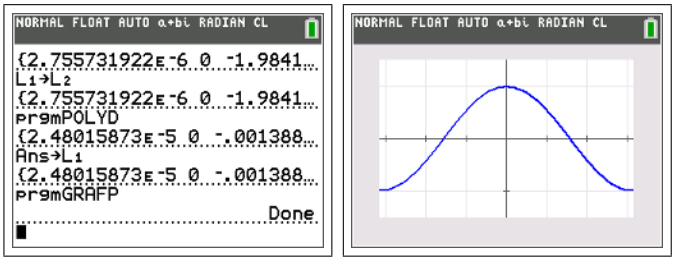
59: We gebruiken onze porgramma's en bekomen volgende antwoorden.

1. De veelterm $S(x)$ ziet er op dit interval uit als de functie $\sin x$.

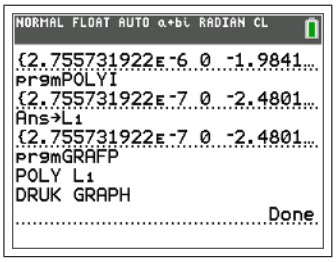


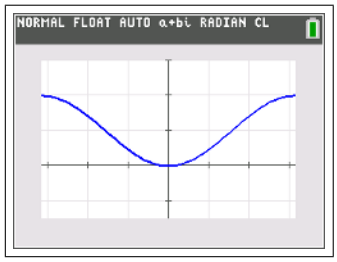
   Zorg ervoor dat je L_1 tijdelijk wegschrijft in L_2, zodat je ze later in de oefening kan terugvinden.

2. Wanneer we $S(x)$ afleiden, krijgen we een andere veelterm die zeer sterk op de functie $\cos x$ lijkt, hetgeen we kunnen verwachten omdat $(\sin x)' = \cos x$.

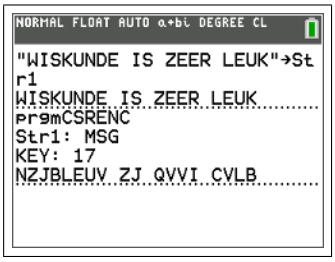


3. Wanneer we $S(x)$ integreren, krijgen we een andere veelterm die qua vorm wel op de functie $\cos x$ lijkt maar hoger ligt ten opzichte van de x-as. Dit komt omdat primitieven gelijk zijn op een constante na.

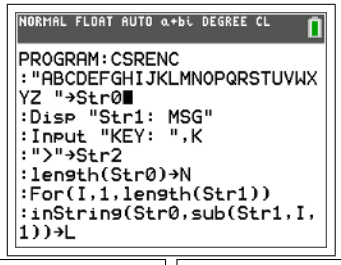


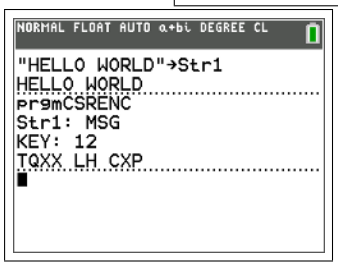


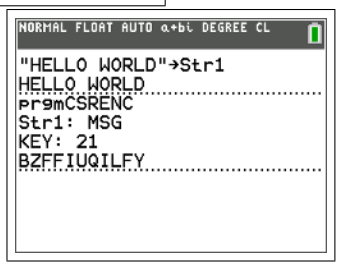
67:



70: Voeg aan het alfabet de spatie toe:







Door deze verandering lijkt het soms of de boodschap uit meerdere woorden, of uit één enkel woord bestaat. De zender en de ontvanger moeten naast de sleutel dus eigenlijk ook een alfabet afspreken.

72: Dit programma is hetzelfde als dat voor de Caesar-code, alleen wordt nu een `for`-lus gebruikt om alle mogelijke sleutels af te gaan. Telkens verschijnt dan het ontcijferde bericht. Eén van deze berichten zal vlot leesbaar zijn.

74: JMCVS YWIVH

76: De regel `inString(Str0,sub(Str3,T,1))-1->K` moet vervangen worden door `N-inString(Str0,sub(Str3,T,1))+1->K`.

78: ROODKAPJE EN DE WOLF, Roald Dahl, Gruwelijke rijmen, Fontein, 1982

79: Indien het bericht CAZ bekomen wordt, is het schema bijvoorbeeld:

$$AAA \longrightarrow \text{ storing } \longrightarrow CAZ$$

Er moeten dus 2 fouten opgetreden zijn. De 3-repetitie-code is dus een foutverbeterende code die in staat is om één fout te verbeteren en te detecteren of twee fouten te detecteren.

80: In dit geval zijn twee fouten opgetreden maar zal de 3-repetitie-code dit niet detecteren. Er zal foutief een bericht Q teruggevonden worden.

81: Detectie en correctie van één fout:

$$AAAA \longrightarrow \text{ storing } \longrightarrow ADAA$$

Detectie en correctie van 2 fouten:

$$AAAA \longrightarrow \text{ storing } \longrightarrow ADXA$$

Detectie van 2 fouten zonder of met verkeerde correctie:

$$AAAA \longrightarrow \text{ storing } \longrightarrow ADDA$$

Detectie van 3 fouten:

$$AAAA \longrightarrow \text{ storing } \longrightarrow ADFX$$

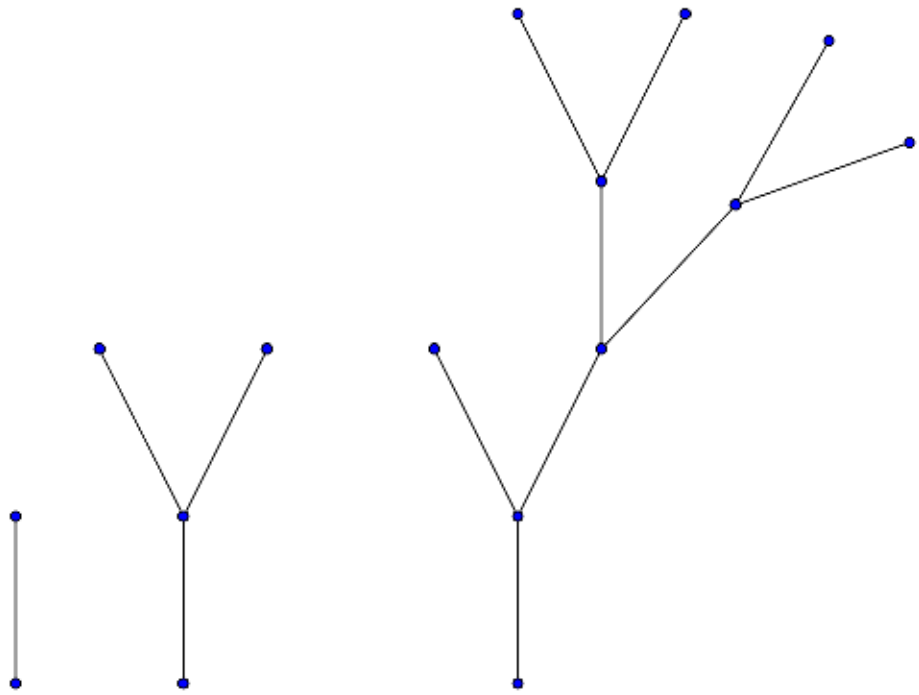
89: Het resultaat zijn twee vierkanten die aan één hoekpunt samenhangen.

91: driehoek: $0, 120, 0, 120, 0$
rechthoek: $0, 0, 0, 90, 0, 0, 90, 0, 0, 0, 90, 0, 0$

92: Enkel rechthoekige driehoeken waarbij de lengte van de zijden Pythagorische tripels zijn, zullen getekend kunnen worden. Bovendien zal men minstens één van de scherpe hoeken moeten berekenen.

94: $L[-L][+L[-L]L]$ is een mogeljike beschrijving. Een andere volgorde van de takken geeft $L[+L[-L]L][-L]$.

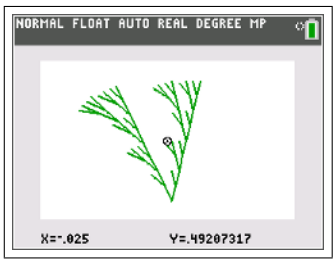
95: Een mogelijke voorstelling van de plant:



97: Het groeipatroon wordt gegeven door $L[+L]L$. Het lijkt een beetje onnatuurlijk want de plant groeit slechts langs één kant.

98: De plant groeit in dat geval schuin.

99: De plant groeit nu vanuit twee schuine takjes. Je kan nu wel geen 5 stappen meer berekenen.



100: Het probleem is dat er in het groeipatroon twee verschillende hoeken staan. De symbolen + en − volstaan dus niet. Als we deze aanpassen

door $+\alpha$ en $-\alpha$ waarbij α de grootte van de hoek is. Het groeipatroon wordt dan $L \rightarrow L[+20°L] - 5°L$.

101: Het groeipatroon van deze plant is $L \rightarrow L[+L][-L]$, waarbij de hoek 20° gekozen werd.

102: Er werd een hoek van $+10°$ en één van $-30°$ gebruikt.

103: De stam werd verlengd. Het groeipatroon is $L \rightarrow LL[+L][-L]$ geworden.

In dit cahier gaan we een aantal onderwerpen aanraken die geschikt zijn als onderzoekscompetenties wiskunde/wetenschappen in het ASO. Elk hoofdstuk kan afzonderlijk door één leerling of in kleine groepjes doorgenomen worden. De onderwerpen komen uit de kruisbestuiving van wiskunde en technologie. In de eerste twee hoofdstukken wordt nagegaan hoe technologie kan helpen om wiskundige problemen aan te pakken. De volgende hoofdstukken gaan dieper in op een aantal voorbeelden waarbij de wiskunde de technologie ter hulp snelt.

DIDIER DESES is leerkracht wiskunde aan het Koninklijk Atheneum Koekelberg en geeft les aan de Wetenschappelijke (5u wisk/week) en de Latijnse richtingen (3u wisk/week). Hij is tevens wetenschappelijk medewerker aan de Vrije Universiteit Brussel.

Augustus 2015

TEXAS INSTRUMENTS