|  |  |
| --- | --- |
| **STEP** | **RESULT** |
|  |  |
| ***ValidateUserInput****()* |  |
| Read userInput |  |
| While userInput is valid |  |
| Call ***ProcessUserInput()*** | 2 |
| Else Return |  |
|  |  |
| ***ProcessUserInput****()* |  |
| Case IserInputof |  |
| UserInput = 1 : probability = 0.20; dof = 6; | 0.20  6 |
| UserInput = 2: probability = 0.45; dof = 15; |  |
| UserInput = 3: probability = 0.495; dof = 4; |  |
|  |  |
| **// First DLD of FindValueOfTMethod** |  |
|  |  |
| ***FindValueOfT()*** |  |
|  |  |
| Tinterval = 1.0 | 1.0 |
| Delta = 0.5 | 0.5 |
| ErrorPrecision = 0.0000001 | 0.0000001 |
| Call ***CalculateProbability()*** *return probabilityFirstIteration* | ***0.3320*** |
| While (Math.Sign(probability – probabilityFirstIteration) == Math.Sign(errorPrecision) | 1. 0.20 – 0.3320 <0 ; errorPrecision >0 => will not enter the while loop |
| { |  |
| If (probability > probabilityFirstIteration) |  |
| { |  |
| Tinterval = tInterval + delta |  |
| Call ***CalculateProbability()*** |  |
| } |  |
| Else if (probability < probabilityFirstIteration) |  |
| { |  |
| Tinterval = tInterval - delta |  |
| Call ***CalculateProbability()*** |  |
| } |  |
| } |  |
| Delta = delta/2; | 1. delta = 0.5/2 = 0.25 **And it will exit => without even trying the next tInterval value (Defect #44)** |
| Return; |  |
|  |  |
| **// Second DLD of FindValueOfTMethod** |  |
|  |  |
| ***FindValueOfT()*** |  |
|  |  |
| Tinterval = 1.0 | 1.0 Try fix defect 45 and move this out of the method |
| Delta = 0.5 | 0.5 Try fix defect 45 and move this out of the method |
| ErrorPrecision = 0.0000001 | 0.0000001 |
| Call ***CalculateProbability()*** *return probabilityFirstIteration* | 1. 0.3320  5. 0.1928 |
| While (Math.Sign(probability – probabilityFirstIteration) <> Math.Sign(errorPrecision) | 1. 0.20 – 0.3320 < 0 – negative math sign; errorPrecision > 0 => the control flow will enter the while loop  2. 0.20 – 0.2592 < 0 – negative math sign; errorPrecision > 0 => the control flow will enter the while loop  3. 0.20 – 0.2293 < 0 – negative math sign; errorPrecision > 0 => the control flow will enter the while loop  4. 0.20 – 0.2029 < 0 – negative math sign; errorPrecision > 0 => the control flow will enter the while loop  5. 0.20 – 0.1928 > 0 – positive sign; errorPrecision > 0 => the control flow will Not enter the while loop  **And it will exit => without even trying the next tInterval value (Defect #45)** |
| { |  |
| Delta = delta/2 | 1. Delta = 0.5/2 = 0.25  2. Delta = 0.25/2 = 0.125  3. Delta = 0.125/2 = 0.0625  4. Delta = 0.0625/2 = 0.03125 |
| If (probability > probabilityFirstIteration) | 1. 0.20 > 0.3320 : False => the control flow will go to the **Else** condition  2. 0.20 > 0.2592 : False => the control flow will go to the **Else** condition  3. 0.20 > 0.2293: False => the control flow will go to the **Else** condition  4. 0.20 > 0.2029 : False => the control flow will go to the **Else** condition |
| { |  |
| Tinterval = tInterval + delta |  |
| Call ***CalculateProbability()*** |  |
| } |  |
| Else if (probability < probabilityFirstIteration) | 1. 0.20 < 0.3320: True  2. 0.20 < 0.2592 : True  3. 0.20 < 0.2225 : True  4. 0.20 < 0.2225 : True |
| { |  |
| Tinterval = tInterval - delta | 2. tInterval = 1.0 – 0.25 = 0.75  3. tInterval = 0.75 – 0.125 = 0.625  4. tInterval = 0.625 – 0.0625 = 0.5625  5. tInterval = 0.5625 – 0.03125 = 0.53125 |
| Call ***CalculateProbability()*** | 1. 0.2592  2. 0.2293  3. 0.2029  4. 0.1928 |
| } |  |
| } |  |
|  |  |
|  |  |
|  |  |
| Return; |  |
|  |  |
| **// Third DLD of FindValueOfTMethod** |  |
|  |  |
| ***FindValueOfT()*** |  |
|  |  |
| Tinterval = 1.0 |  |
| Delta = 0.5 |  |
| ErrorPrecision = 0.0000001 | 0.0000001 |
| Call ***CalculateProbability()*** *return probabilityFirstIteration* | 1. 0.3320 |
|  |  |
| ***While (NOT !(probability – probability <= E))*** | 6. 0.20 – 0.2029 < E : True  7. 0.20 – 0.1979 < E : True  8. 0.20 – 0.2029 < E : True  9. 0.20 – 0.2004 < E : True  10. 0.20 – 0.1992< E: True  11. 0.20 – 0.2004 < E : True  12. 0.20 – 0.1998 < E : True  13. 0.20 – 0.2004 < E : True  14. 0.20 – 0.2001 < E :True  15. 0.20 – 0.1999 < E : True  16. 0.20 – 0.2001 < E :True  17. 0.20 – 0.20002765 < E: True  18. 19998840 // skip the verification below since it's clear it'll return #19 equal to #17  19. 0.20002765  20. 0.19996877 // skip the verification below since it's clear it'll return #21 equal to #19  21. 0.20002765 |
|  |  |
| **IF**(Math.Sign(probability – probabilityFirstIteration) <> Math.Sign(errorPrecision) | 1. 0.20 – 0.3320 < 0 – negative math sign; errorPrecision > 0 => the control flow will enter This If condition  2. 0.20 – 0.2592 < 0 – negative math sign; errorPrecision > 0 => the control flow will enter This If condition  3. 0.20 – 0.2293 < 0 – negative math sign; errorPrecision > 0 => the control flow will enter This If condition  4. 0.20 – 0.2029 < 0 – negative math sign; errorPrecision > 0 => the control flow will enter This If condition  5. 0.20 – 0.1928 > 0 – positive sign; errorPrecision > 0 => the control flow will NOT enter This If condition but its corresponding ELSE condition  7. 0.20 – 0.2029 > 0 - positive math sign; errorPrecision > 0 => the control flow will Not enter This If condition  8. 0.20 – 0.1979 > - positive will not enter this IF  9. 0.20 – 0.2029 > 0 - positive math sign; errorPrecision > 0 => the control flow will Not enter This If condition  10. 0.20 – 0.2004 < 0 – negative math sign; errorPrecision > 0 => the control flow will Enter this IF condition  11. 0.20 – 0.1992 > 0 => Else  12. 0.20 – 0.2004 < 0 – negative math sign; errorPrecision > 0 => the control flow will Enter this IF condition  13. 0.20 – 0.1998 > 0 => Else  14. 0.20 – 0.2004 < 0 – negative math sign; errorPrecision > 0 => the control flow will Enter **this IF** condition  15. 0.20 - 0.2001 < 0 – negative math sign; errorPrecision > 0 => the control flow will Enter **this IF** condition |
|  | 16. 0.20 – 0.1999 > 0 => Else  17. 0.20 - 0.2001 < 0 – negative math sign; errorPrecision > 0 => the control flow will Enter **this IF**  18. 0.20 - 0.20002765 < 0 – negative math sign; errorPrecision > 0 => the control flow will Enter this IF  20. 0.20 - 0.20002765 < 0 – negative math sign; errorPrecision > 0 => the control flow will Enter this IF  22. 0.20 - 0.20002765 < 0 – negative math sign; errorPrecision > 0 => the control flow will Enter this IF |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
| { |  |
| Delta = delta/2 | 1. Delta = 0.5/2 = 0.25  2. Delta = 0.25/2 = 0.125  3. Delta = 0.125/2 = 0.0625  4. Delta = 0.0625/2 = 0.03125  7. Delta = 0.03125/2 = 0.015625  9. Delta = 0. 015625/2 = 0.0078125  10. Delta = 0.0078125/2 = 0.00390625  12. Delta = 0.00390625/2 = 0.001953125  14. Delta = 0.001953125/2 = 0.0009765625  15. Delta = 0.0009765625/2 = 0.00048828125  17. Delta = 0.00048828125/2 = 0.000244140625  18. Delta = 0.000244140625/2 = 0.0001220703125  20. Delta = 0.0001220703125/2 = 0.00006103515625 |
|  | 22. Delta = 0.00006103515625/2 = 0.000030517578125 |
|  |  |
|  |  |
|  |  |
| If (probability > probabilityFirstIteration) | 1. 0.20 > 0.3320 : False => the control flow will go to the **Else** condition  2. 0.20 > 0.2592 : False => the control flow will go to the **Else** condition  3. 0.20 > 0.2293: False => the control flow will go to the **Else** condition  4. 0.20 > 0.2029 : False => the control flow will go to the **Else** condition  7. 0.20 > 2029: False => the control flow will go to the **Else** condition  9. 0.20 > 2029: False => the control flow will go to the **Else** condition  10. 0.20 > 0.2004: False => the control flow will go to the **Else** condition  12. 0.20 > 0.2004: False => the control flow will go to the **Else** condition  14. 0.20 > 0.2004: False => the control flow will go to the **Else** condition  15. 0.20 > 0.2001: False => the control flow will go to the **Else** condition  17. 0.20 > 0.2001: False => the control flow will go to the **Else** condition  18. 0.20 > 0.20002765 : False => the control flow will go to the **Else** condition |
| { |  |
| Tinterval = tInterval + delta |  |
| Call ***CalculateProbability()*** |  |
| } |  |
| Else if (probability < probabilityFirstIteration) | 1. 0.20 < 0.3320: True  2. 0.20 < 0.2592 : True  3. 0.20 < 0.2225 : True  4. 0.20 < 0.2029 : True  7. 0.20 < 0.2029 : True  9. 0.20 < 0.2029 : True  10. 0.20 < 0.2004 : True  12. 0.20 < 0.2004 : True  14. 0.20 < 0.2004 : True  15. 0.20 < 0.2001 : True  18. 0.20 < 0.20002765 :True |
| { |  |
| Tinterval = tInterval - delta | 2. tInterval = 1.0 – 0.25 = 0.75  3. tInterval = 0.75 – 0.125 = 0.625  4. tInterval = 0.625 – 0.0625 = 0.5625  5. tInterval = 0.5625 – 0.03125 = 0.53125  7. tInterval = 0.5625 – 0.015625 = 0.546875  9. tInterval = 0.5625 – 0.0078125 = 0.5546875  10. tInterval = 0.5546875 – 0.00390625 = 0.55078125  12. tInterval = 0.5546875 – 0.001953125 = 0.552734375  14. tInterval = 0.5546875 – 0.0009765625 = 0.5537109375  15. tInterval = 0.5537109375 – 0.00048828125 = 0.55322265625  17. tInterval = 0.5537109375 – 0.000244140625 = 0.553466796875  18. tInterval = 0.553466796875 – 0.0001220703125 = 0.5533447265625  20. tInteval = 0.5533447265625 – 0.00006103515625 = 0.55328369140625  22. tInterval = 0.55328369140625 - 0.000030517578125 …......**. *Terminate Design Verification. Design Proved Working!*** |
| Call ***CalculateProbability()*** | 1. 0.2592  2. 0.2293  3. 0.2029  4. 0.1928  7. 0.1929  9. 0.2004  10. 0.1992  12. 0.1998  14. 0.2001  15. 0.1999  17. 0.20002765  18. 19998840  19. 0.20002765  20. 0.19996877 |
| } |  |
| Else if (Math.Sign(probability – probabilityFirstIteration) == Math.Sign(errorPrecision) | 5. 0.20 – 0.1928 > 0– positive sign; errorPrecision > 0 => the control flow will enter This ELSE condition  8. 0.20 – 0.1979 > 0 – positive sign same as ErrorPrecision, the control flow will enter this ELSE condition  11. 0.20 – 0.1992 > 0 – positive sign same as ErrorPrecision, the control flow will enter this ELSE condition  13. 0.20 – 0.1998 > 0 – positive sign same as ErrorPrecision, the control flow will enter this ELSE condition  16. 0.20 – 0.1999 > 0 – positive sign same as ErrorPrecision, the control flow will enter this ELSE condition |
| If (probability > probabilityFirstIteration) | 5. 0.20 > 0.1928 : True – the control flow will enter this IF condition  8. 0.20 > 0.1979 : True – the control flow will enter this IF condition  11. 0.20 > 0.1992 : True – the control flow will enter this IF condition  13. 0.20 > 0.1998 : True - the control flow will enter this IF condition  16. 0.20 > 0.1999 : True - the control flow will enter this IF condition |
| { |  |
| Tinterval = tInterval + delta | 6. tInterval = 0.53125 + 0.03125 = 0.5625  8. tInterval = 0.546875 + 0.015625 = 0.5625  11. tInterval = 0.55078125 + 0.00390625 = 0.5546875  13. tInterval = 0.552734375 + 0.001953125 = 0.5546875  16. tInterval = 0.55322265625 + 0.00048828125 = 0.5537109375 |
| Call ***CalculateProbability()*** | 6. 0.2029  8. 0.2029  11. 0.2004  13. 0.2004  16. 0.2001 |
| } |  |
| Else if (probability < probabilityFirstIteration) |  |
| { |  |
| Tinterval = tInterval - delta |  |
| Call ***CalculateProbability()*** |  |
|  |  |
| } |  |
| } |  |
| Return; |  |
| } |  |
|  |  |
| } |  |
| ***GetRangeValues****()* |  |
| For I = 0; I <numberOfsegments+1; I ++ |  |
| WidthOfI[i] = i\*tInterval/numberOfSegments |  |
|  |  |
| ***CalculateGamma****(dof)* |  |
| If dof == 1 return 1 |  |
| If dof == 0.5 return |  |
| Return (dof-1)\*CalculateGamma(dof+1) |  |
|  |  |
| ***CalculateConstant****()* |  |
| Return CalculateGamma((dof + 1) / 2) /\*CalculateGamma(dof/2) |  |
|  |  |
| **CalculateResult**() |  |
| For I = 0; I < numberOfSegments +1; i++ |  |
| FunctionOfWidth[i] = CalculateConstant(dof)\* |  |
|  |  |
| ***CalculateSumOfTerms****()* |  |
| SumOfEndTerms = funtionOfWidth[0] + functionOfWidth[tInterval] |  |
|  |  |
| For I = 1; I < numberOfSegments; I+=2 |  |
| sumOfOddTerms[i] = functionOfWidth[i]\*4 |  |
| SumOfTerms += sumOfOddTerms[i] |  |
|  |  |
| For J = 2; J < numberOfSegments; J +=2 |  |
| sumOfEvenTerms[j] = functionOfWidth[i]\*2; |  |
| SumOfTerms += sumOfEvenTerms |  |
|  |  |
| SumOfTerms += sumOfEndTerms; |  |
|  |  |
| ***CalculateProbability()*** |  |
|  |  |
| Call ***CalculateSumOfTerms()*** |  |
|  |  |
| Multiplier = tInterval/numberOfSegments\*3 |  |
|  |  |
| ProbabilityFirstIteration = multiplier\*sumOfTerms |  |
|  |  |
|  |  |