ภาคผนวก ข คู่มือการใช้ซอฟต์แวร์ Gambit และ Fluent บนระบบปฏิบัติการ LINUX

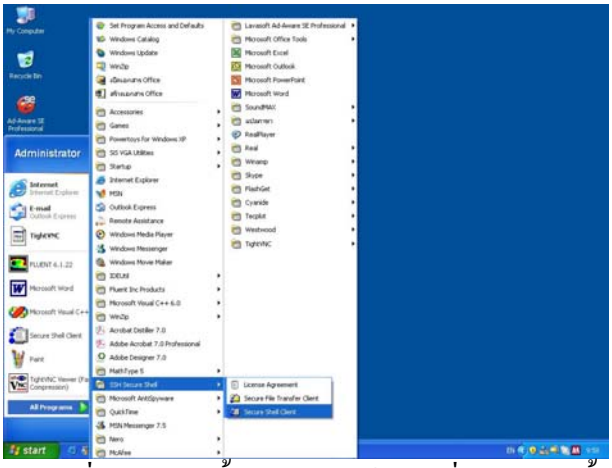
ขั้นตอนเริ่มต้นก่อนใช้งาน

ในคู่มือการใช้ในงานวิจัยนี้ได้ทำบนเครื่อง Helios ที่ APSTF Lab. จึงอ้างอิงวิธีใช้จากเครื่อง Helios แต่ระบบวิธีใช้หลักๆจะเหมือนกันทุกเครื่อง

1. ขอ Account ในการใช้เครื่อง Helios ที่ http://apstf.cpe.ku.ac.th/
2. ทำการติดตั้งโปรแกรม SSH SecureShellClient และ Tightvnc โดยทำการ Download ได้ที่ http://apstf.cpe.ku.ac.th/index.php?option=com_content&task=view&id=29&Itemid=37

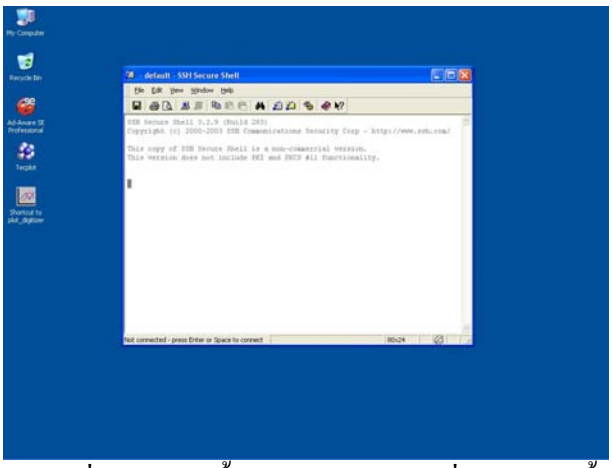
วิธีการเข้าเครื่อง Helios ผ่าน SecureShell

1) ไปที่ Start → all program → SSH SecureShellClient → SecureShellClient



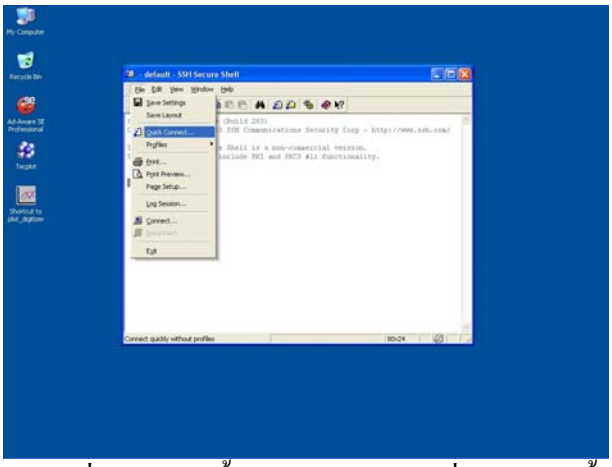
ภาพผนวกที่ ข1 แสดงขั้นตอนการใช้งานเครื่อง Helios ขั้นที่ 1

2) จะได้หน้าจอทำงานของSecureShellClient



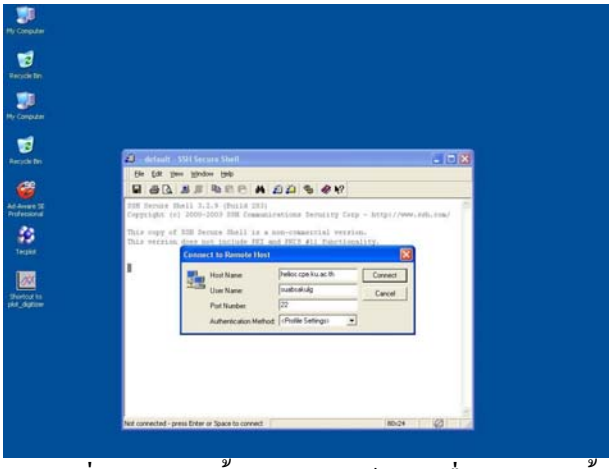
ภาพผนวกที่ ข2 แสดงขั้นตอนการใช้งานเครื่อง Helios ขั้นที่ 2

3) เลือก flie → Quick connect



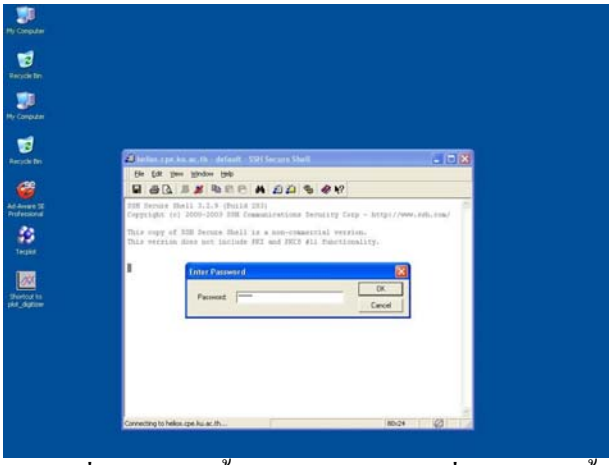
ภาพผนวกที่ ข3 แสดงขั้นตอนการใช้งานเครื่อง Helios ขั้นที่ 3

4) ที่ Host Name พิมพ์ helios.cpe.ku.ac.th ที่ User Nome พิมพ์ ชื่อ Account ที่ขอได้



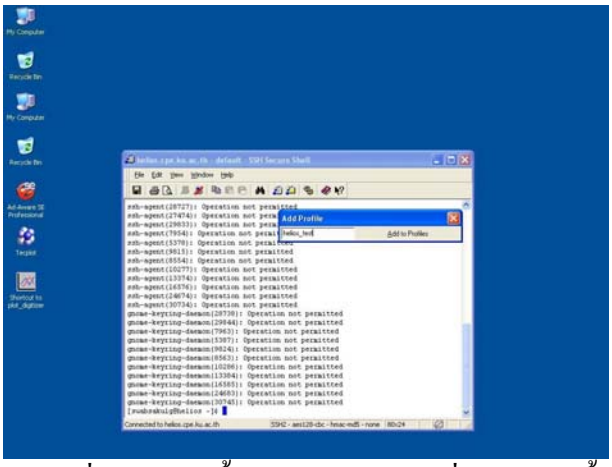
ภาพผนวกที่ ข4 แสดงขั้นตอนการใช้งานเครื่อง Helios ขั้นที่ 4

5) ใส่ Password



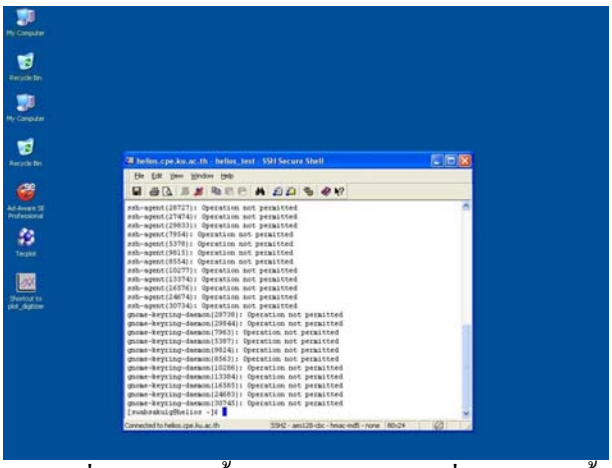
ภาพผนวกที่ ข5 แสดงขั้นตอนการใช้งานเครื่อง Helios ขั้นที่ 5

6) ผู้ใช้สามารถตั้งค่าให้ Secureshell สามารถจำชื่อ Account ที่ใช้นี้ได้ โดยตั้งชื่อที่ Profile



ภาพผนวกที่ ข6 แสดงขั้นตอนการใช้งานเครื่อง Helios ขั้นที่ 6

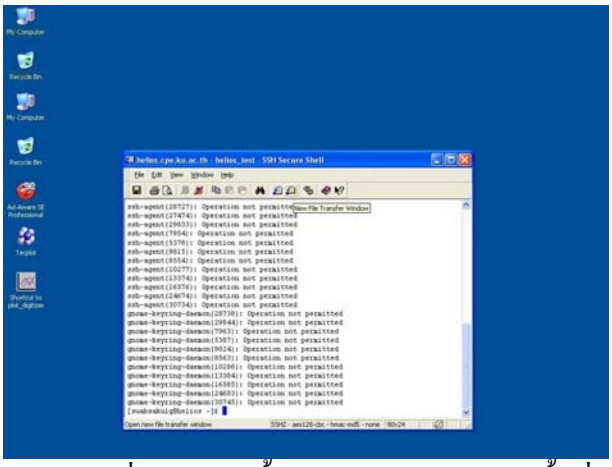
7) หน้าจอการทำงานของ Secureshell โดยเริ่มต้นผู้ใช้จะอยู่ใน ไดเร็กทอรี่ home/ชื่อ Account เสมอ



ภาพผนวกที่ ข7 แสดงขั้นตอนการใช้งานเครื่อง Helios ขั้นที่ 7

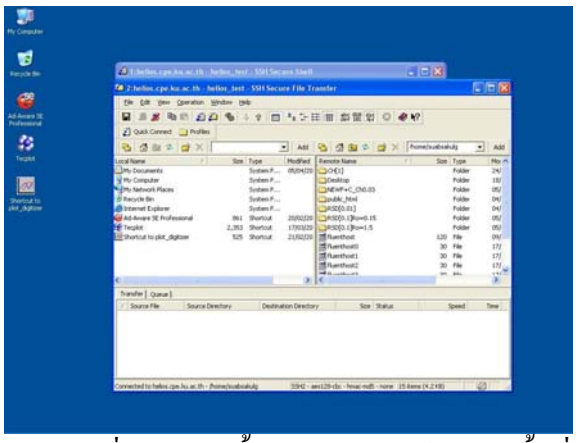
การโอนย้าย File ต่างๆจากเครื่อง PC ของผู้ใช้เพื่อนำมาใช้ในเครื่อง Helios

1. เลือกที่ New file transfer window



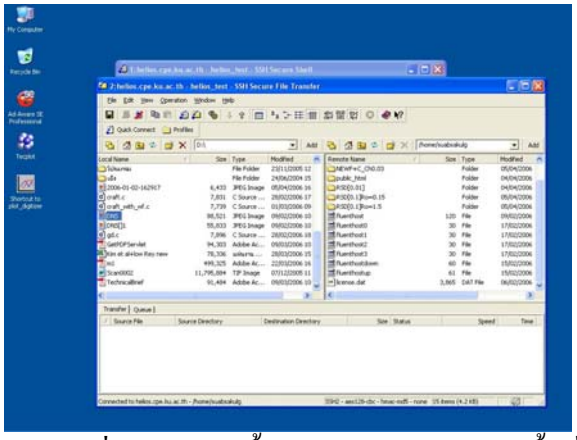
ภาพผนวกที่ ข8 แสดงขั้นตอนการโอนย้าย File ขั้นที่ 1

2. จะได้หน้าจอ SSH secure file transfer ในตำแหน่งของ Folder ด้านซ้ายมือจะเป็น Folder ในส่วนของเครื่อง PC ที่ผู้ใช้ใช้อยู่ ส่วนด้านขวามือนั้นเป็น folder ในส่วนของ home/ชื่อ Account ภายในเครื่อง Helios



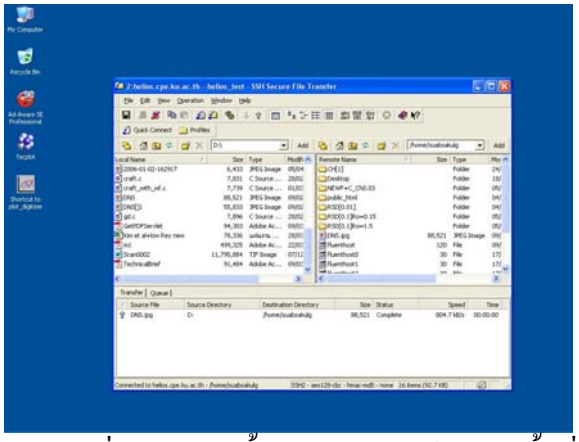
ภาพผนวกที่ ข9 แสดงขั้นตอนการโอนย้าย File ขั้นที่ 2

3. วิธีการโอนย้าย file ไปยัง Helios นั้นทำได้โดยใช้ mouser กดเลือก file ที่ต้องการค้างไว้จากนั้นลากไปไว้ที่หน้าจอฝั่งด้านขวาซึ่งเป็นส่วนของเครื่อง Helios ได้ทันที



ภาพผนวกที่ ข10 แสดงขั้นตอนการโอนย้าย File ขั้นที่ 3

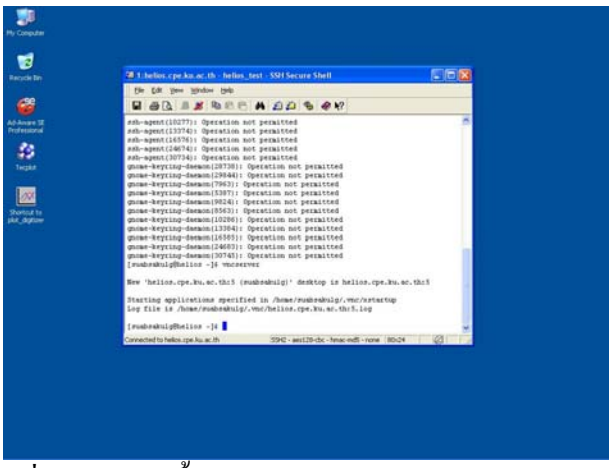
4. จากตัวอย่างสามารถย้าย file ที่ชื่อ DNS ไปอยู่ที่เครื่อง Helios ได้แล้ว



ภาพผนวกที่ ข11 แสดงขั้นตอนการโอนย้าย File ขั้นที่ 4

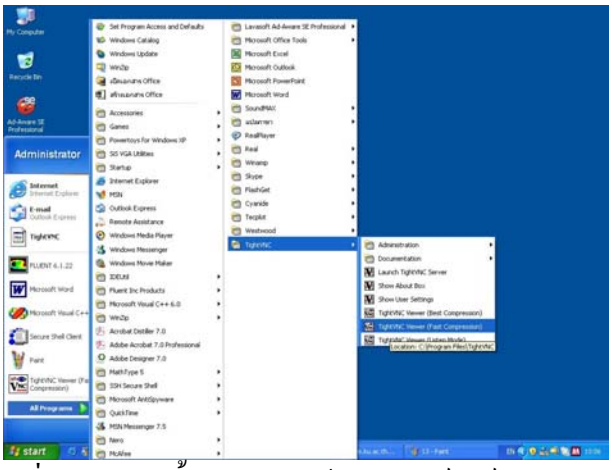
วิธีการเปิดใช้หน้าต่างทำงานเพื่อเปิดโปรแกรมต่างๆภายในเครื่อง Helios ผ่านซอฟต์แวร์ TightVNC

1. ที่หน้าจอ Secureshell พิมพ์ vncserver จะปรากฏ section ที่ผู้ใช้สามารถเข้าไปใช้ได้จาก ตัวอย่างได้แก่ helios.cpe.ku.ac.th:5 ดังนั้น section ที่ผู้ใช้เข้าได้คือ section 5



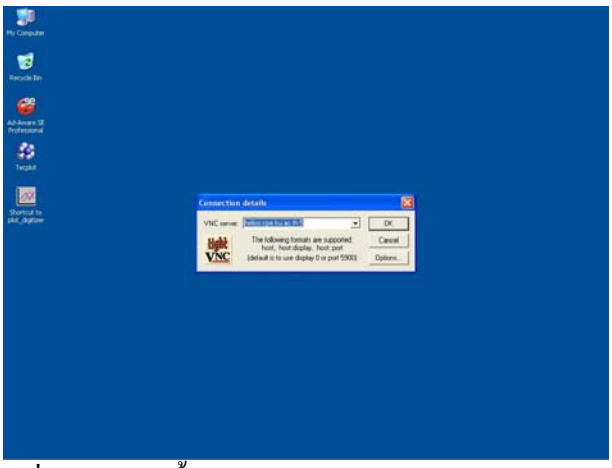
ภาพผนวกที่ ข12 แสดงขั้นตอนการใช้งานซอฟต์แวร์ TightVNC ขั้นที่ 1

2. หลังจากนั้นไปเปิดโปรแกรม TightVNC viewer (fast compression)



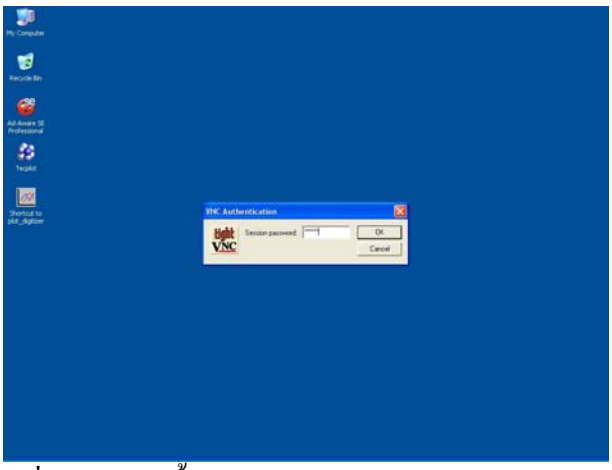
ภาพผนวกที่ ข13 แสดงขั้นตอนการใช้งานซอฟต์แวร์ TightVNC ขั้นที่ 2

3. ทำการใส่ชื่อแล้วตามด้วยหมายเลข section ดังนี้ helios.cpe.ku.ac.th:5



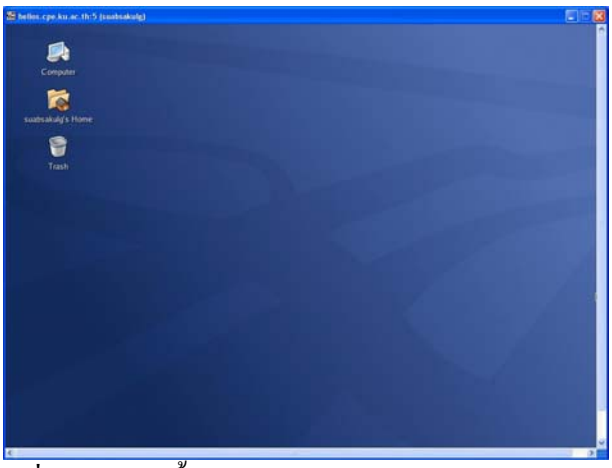
ภาพผนวกที่ ข14 แสดงขั้นตอนการใช้งานซอฟต์แวร์ TightVNC ขั้นที่ 3

4. ใส่ค่า Password ซึ่งไม่จำเป็นต้องเหมือนกับ Account ก็ได้ แต่ต้องใช้ Password นี้ในการเปิด TightVNC ตลอดทุกครั้ง



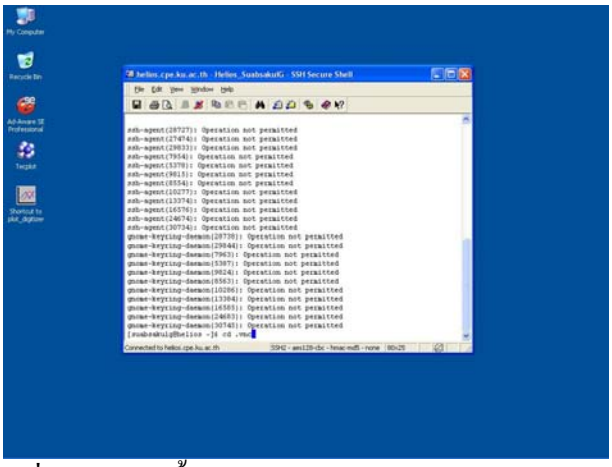
ภาพผนวกที่ ข15 แสดงขั้นตอนการใช้งานซอฟต์แวร์ TightVNC ขั้นที่ 4

5. จะได้หน้าจอปฏิบัติงานของเครื่อง Helios ซึ่งเป็นระบบ LINUX ถ้าหน้าจอของผู้ใช้ไม่เป็นแบบนี้ให้แก้ไขดังต่อไปนี้



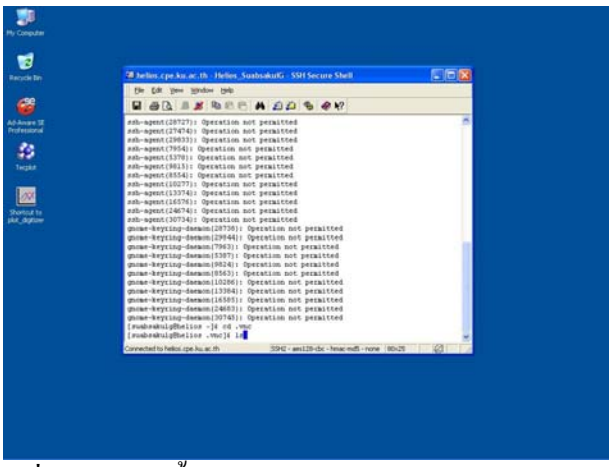
ภาพผนวกที่ ข16 แสดงขั้นตอนการใช้งานซอฟต์แวร์ TightVNC ขั้นที่ 5

6. แก้ไขดังนี้ กลับไปที่หน้าจอ Secureshell แล้วพิม cd .vnc



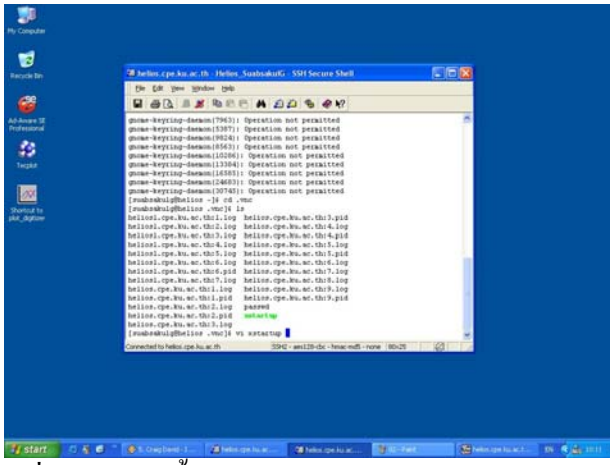
ภาพผนวกที่ ข17 แสดงขั้นตอนการใช้งานซอฟต์แวร์ TightVNC ขั้นที่ 6

7. จากนั้น พิมพ์ ls เพื่อดู File ใน Folder นี้



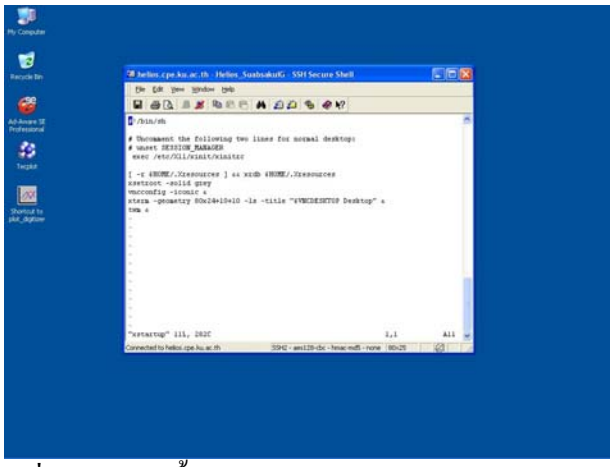
ภาพผนวกที่ ข18 แสดงขั้นตอนการใช้งานซอฟต์แวร์ TightVNC ขั้นที่ 7

8. พิมพ์ vi xstartup



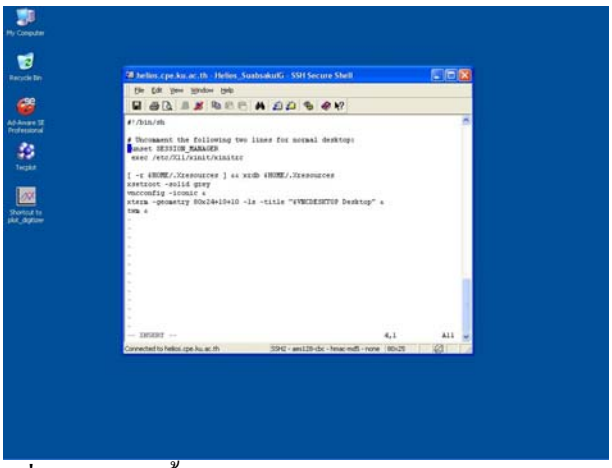
ภาพผนวกที่ ข19 แสดงขั้นตอนการใช้งานซอฟต์แวร์ TightVNC ขั้นที่ 8

9. จะเป็นการเปิด file xstartup



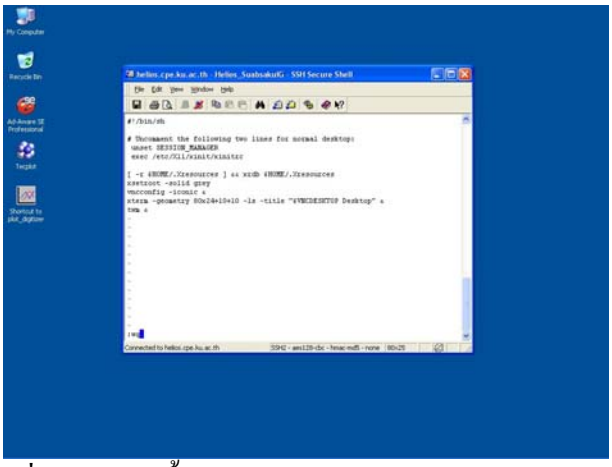
ภาพผนวกที่ ข20 แสดงขั้นตอนการใช้งานซอฟต์แวร์ TightVNC ขั้นที่ 9

10. พิมพ์ i แล้วลบเครื่องหมาย # ที่บรรทัดที่ 4 และ 5 ของ File



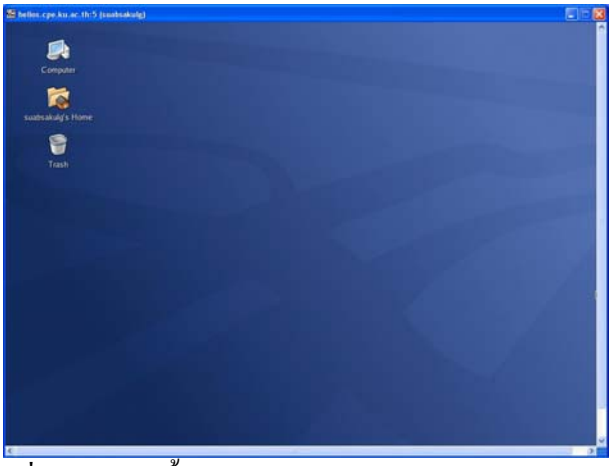
ภาพผนวกที่ ข21 แสดงขั้นตอนการใช้งานซอฟต์แวร์ TightVNC ขั้นที่ 10

11. จากนั้นกดปุ่ม Esc บนแป้นพิมพ์ แล้วพิมพ์ :wq เป็นการ save แล้วออกจาก file xstartup แล้วทำการเข้า TightVNC ใหม่ เพียงเท่านี้หน้าจอปฏิบัติงานบนเครื่อง Helios ก็จะเป็นเหมือนตัวอย่างแล้ว



ภาพผนวกที่ ข22 แสดงขั้นตอนการใช้งานซอฟต์แวร์ TightVNC ขั้นที่ 11

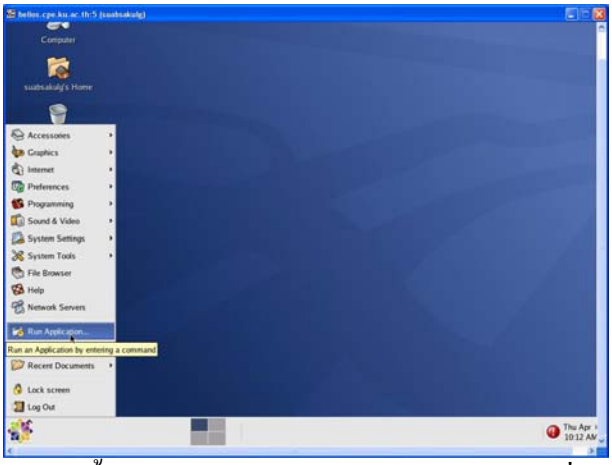
12. เมื่อเข้ามาแล้วเริ่มต้นผู้ใช้จะทำงานอยู่ที่เครื่อง frontend ของเครื่อง Helios



ภาพผนวกที่ ข23 แสดงขั้นตอนการใช้งานซอฟต์แวร์ TightVNC ขั้นที่ 12

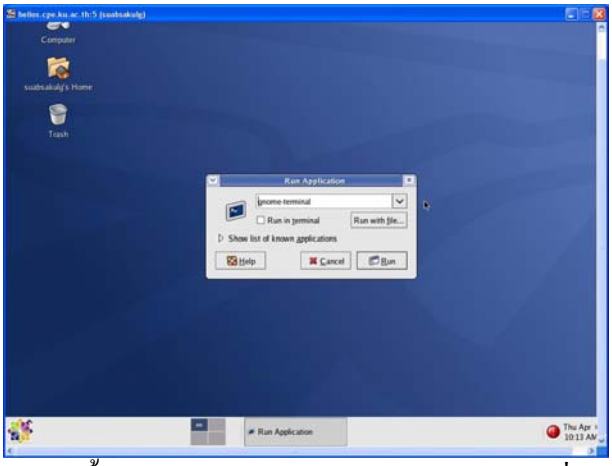
ขั้นตอนการใช้งานซอฟต์แวร์ Fluent บนเครื่อง Frontend

1. เลือก Run application.



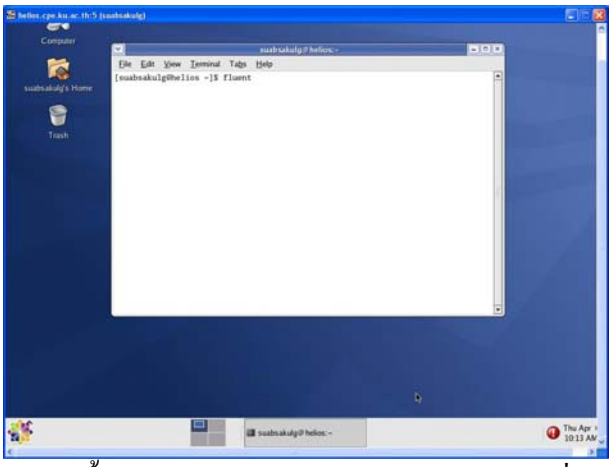
ภาพผนวกที่ ข24 แสดงขั้นตอนการใช้งานซอฟต์แวร์ Fluent บนเครื่อง Frontend ขั้นที่ 1

2. พิมพ์ gnome-terminal → Run



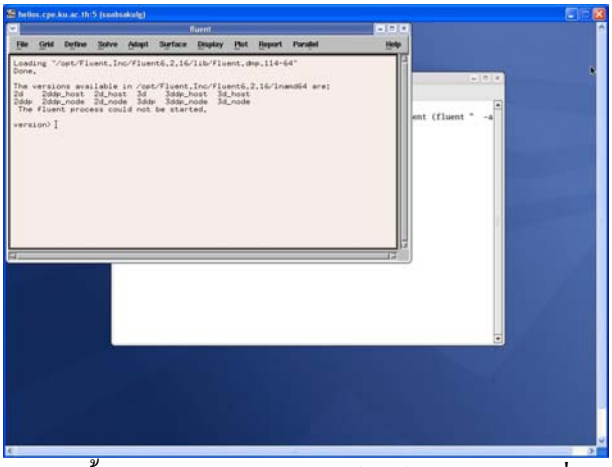
ภาพผนวกที่ ข25 แสดงขั้นตอนการใช้งานซอฟต์แวร์ Fluent บนเครื่อง Frontend ขั้นที่ 2

3. จากนั้น พิมพ์ fluent หรือ gambit



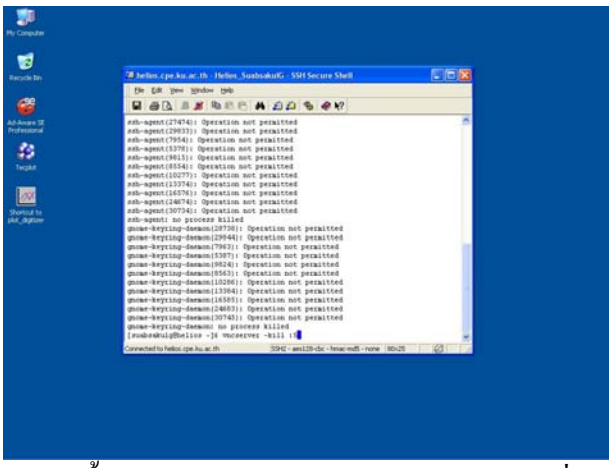
ภาพผนวกที่ ข26 แสดงขั้นตอนการใช้งานซอฟต์แวร์ Fluent บนเครื่อง Frontend ขั้นที่ 3

4. หน้าจอปฏิบัติงานของ Fluent



ภาพผนวกที่ ข27 แสดงขั้นตอนการใช้งานซอฟต์แวร์ Fluent บนเครื่อง Frontend ขั้นที่ 4

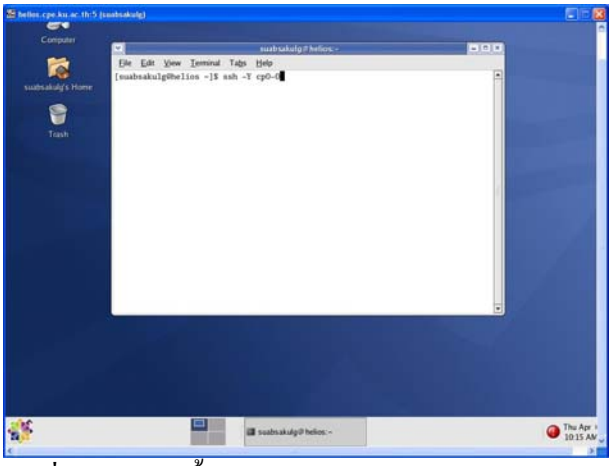
5. เมื่อเลิกใช้แล้วผู้ใช้ต้องมาปิด section ที่เปิดไว้โดยการพิมพ์คำสั่งตามด้วยหมาเลข section ดังนี้ vncserver –kill :5



ภาพผนวกที่ ข28 แสดงขั้นตอนการใช้งานซอฟต์แวร์ Fluent บนเครื่อง Frontend ขั้นที่ 5

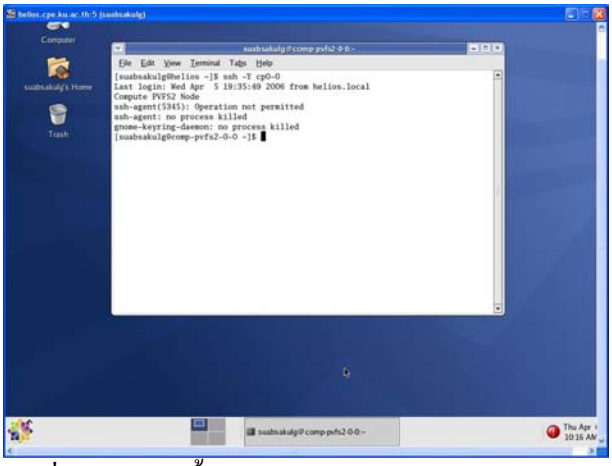
ขั้นตอนการย้ายการทำงานของ Fluent ไปที่เครื่องลูก

1. ที่หน้าจอ genome-terminal พิมพ์คำสั่งไปที่เครื่องลูกตามด้วยชื่อของเครื่องลูกที่ต้องการ ssh –Y cp0-0 ใน Helios มีเครื่องลูกทั้งหมด 4 ตัวสามารถไปที่เครื่องลูกตามชื่อได้ดังต่อไปนี้ cp0-0, cp0-1, cp0-2 และ cp0-3



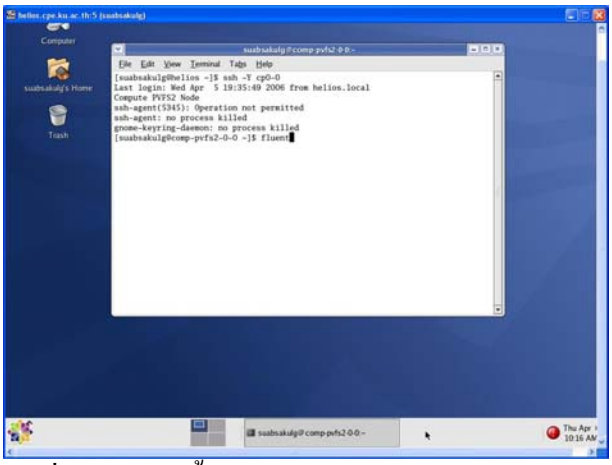
ภาพผนวกที่ ข29 แสดงขั้นตอนการย้ายการทำงานของ Fluent ขั้นที่ 1

2. จะได้หน้าจอการทำงานดังนี้



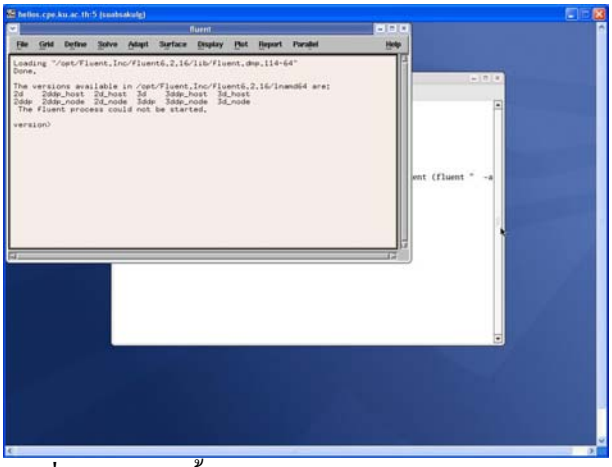
ภาพผนวกที่ ข30 แสดงขั้นตอนการย้ายการทำงานของ Fluent ขั้นที่ 2

3. จากนั้นสามารถสั่งงานใช้ Fluent หรือ gambit ได้ตามปกติ



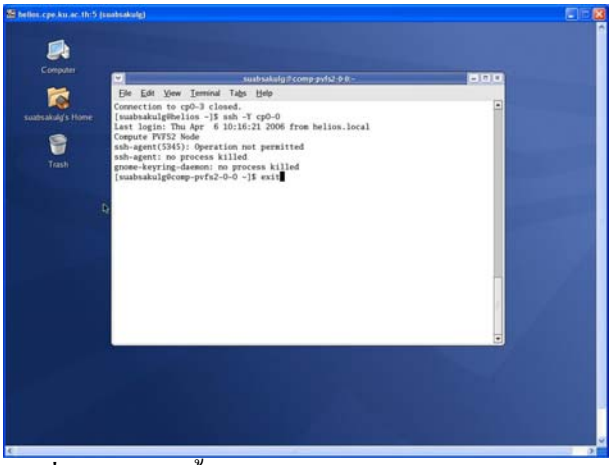
ภาพผนวกที่ ข31 แสดงขั้นตอนการย้ายการทำงานของ Fluent ขั้นที่ 3

4. หน้าจอของ Fluent



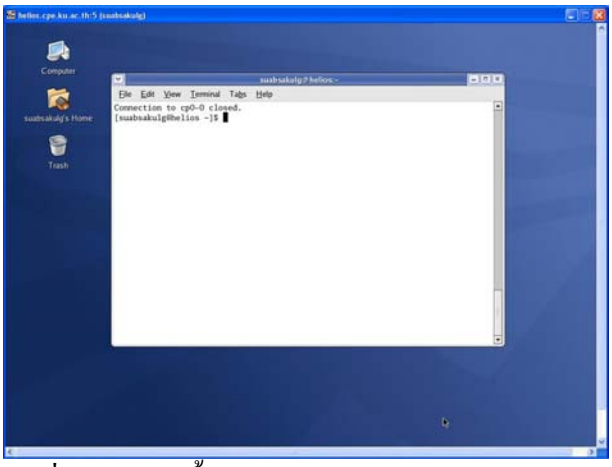
ภาพผนวกที่ ข32 แสดงขั้นตอนการย้ายการทำงานของ Fluent ขั้นที่ 4

5. เมื่อเลิกใช้ Fluent ที่เครื่องลูกต้องทำการปิดโดยพิมพ์ exit



ภาพผนวกที่ ข33 แสดงขั้นตอนการย้ายการทำงานของ Fluent ขั้นที่ 5

6. จัดการปิดเสร็จสิ้น

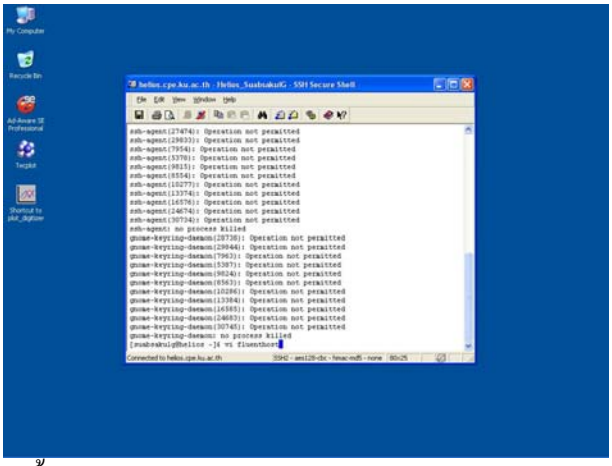


ภาพผนวกที่ ข34 แสดงขั้นตอนการย้ายการทำงานของ Fluent ขั้นที่ 6

ภาคผนวก ค คู่มือการใช้งานซอฟต์แวร์ Fluent แบบขนาน

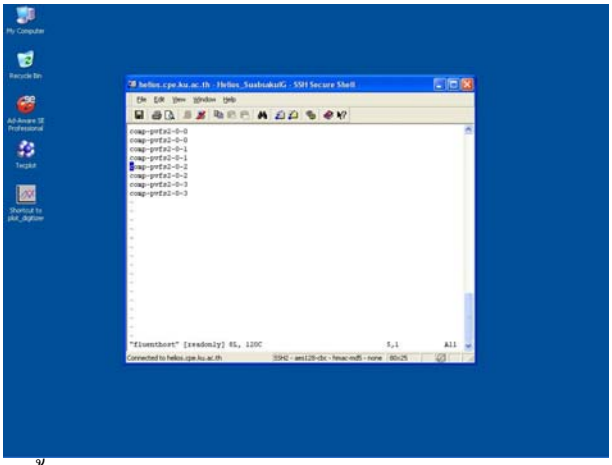
ขั้นตอนการใช้งานซอฟต์แวร์ Fluent แบบขนานบนเครื่อง Helios

1. ที่หน้าจอปฏิบัติการณ์ Secureshell ณ ตำแหน่ง home ไดเร็กทอรี่ของผู้ใช้ ทำการสร้าง file ที่บอกชื่อของเครื่องลูกที่จะใช้ในการประมวลผล โดยการพิมพ์ vi fluenthost



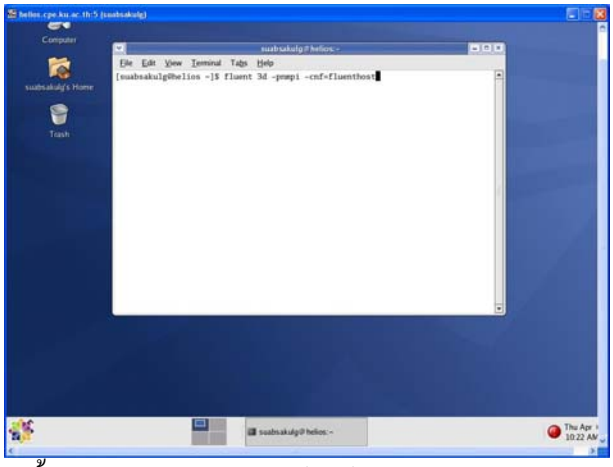
ภาพผนวกที่ ค1 แสดงขั้นตอนการใช้งานซอฟต์แวร์ Fluent แบบขนานบนเครื่อง Helios ขั้นที่ 1

2. เนื่องจากเครื่องลูกของ Helios แต่ละเครื่องมี CPU 2 ตัวดังนั้นจำนวนเครื่องลูกที่จะนำมาร่วมประมวลผลจึงมีได้สูงสุด 8 เครื่อง พิมพ์ชื่อเครื่องลูกดังตัวอย่าง จากนั้นกด Esc แล้วตามด้วย :wq



ภาพผนวกที่ ค2 แสดงขั้นตอนการใช้งานซอฟต์แวร์ Fluent แบบขนานบนเครื่อง Helios ขั้นที่ 2

3. จากนั้นไปที่หน้าจอ gnome-terminal พิมพ์คำสั่ง fluent 3d –pnmpi –cnf=fluenthost คำสั่งแต่ละตัวมีความหมายคือ 3d คือ งานที่ใช้คำนวณเป็นแบบ 3 มิติ ถ้าเป็น 2 มิติ ใช้ 2d ส่วน -pnmpi หมายถึงการเชื่อมต่อ เครื่องลูกโดยใช้ MPI ส่วนสุดท้าย –cnf=fluenthost หมายถึงการกำหนดเครื่องลูก ว่าจะใช้เครื่องไหนในการคำนวณบ้างกำหนดโดยสร้าง File เป็นตัวบอก



ภาพผนวกที่ ค3 แสดงขั้นตอนการใช้งานซอฟต์แวร์ Fluent แบบขนานบนเครื่อง Helios ขั้นที่ 3

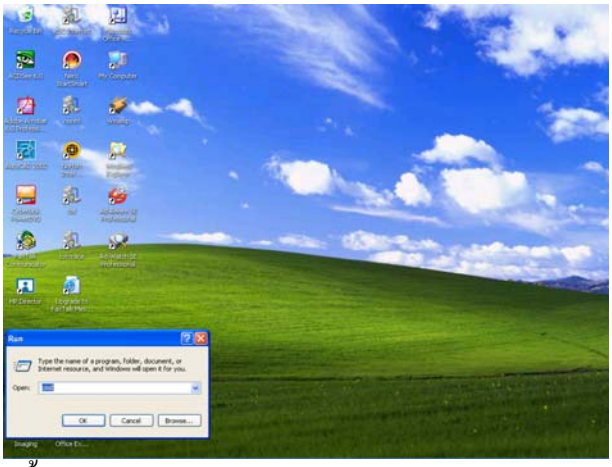
สำหรับตัวอย่างการใช้งานนั้นเมื่อเข้าสู่โปรแกรมไปแล้วจะสามารถใช้งานเหมือนบนระบบปฏิบัติการ Window สามารถดูได้ข้างต้น

ขั้นตอนการใช้งานซอฟต์แวร์ Fluent แบบขนานบนระบบปฏิบัติการ Windows

1. ในการใช้งานต้องเป็น Windows รุ่น 2003 server เท่านั้น เลือก start→run

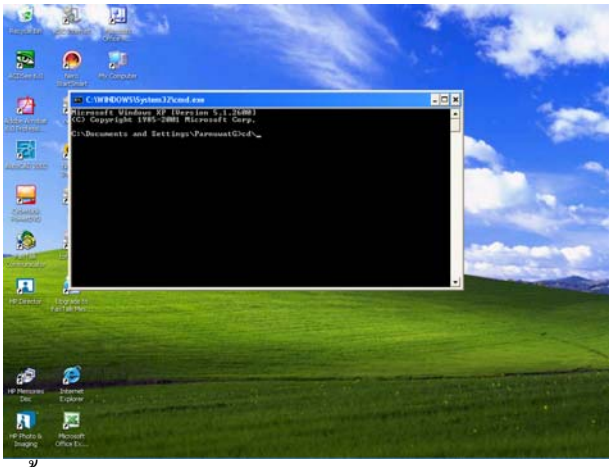
ภาพผนวกที่ ค4 แสดงขั้นตอนการใช้งานซอฟต์แวร์ Fluent แบบขนานบน Windows ขั้นที่ 1

2. พิมพ์ cmd → OK



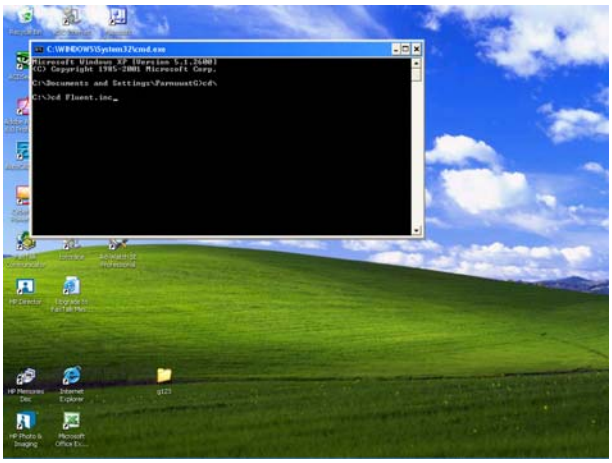
ภาพผนวกที่ ค5 แสดงขั้นตอนการใช้งานซอฟต์แวร์ Fluent แบบขนานบน Windows ขั้นที่ 2

3. พิมพ์ cd\ → enter



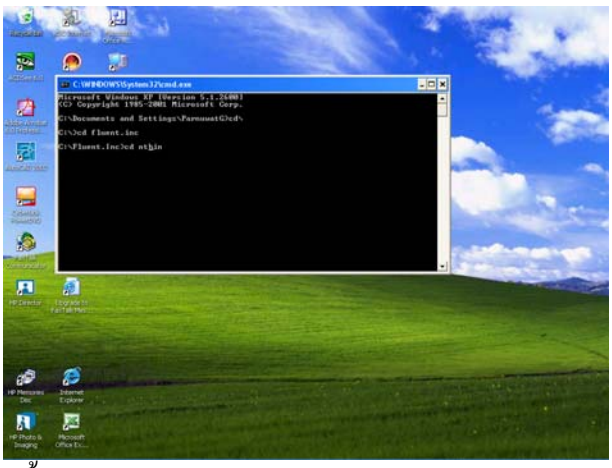
ภาพผนวกที่ ค6 แสดงขั้นตอนการใช้งานซอฟต์แวร์ Fluent แบบขนานบน Windows ขั้นที่ 3

4. พิมพ์ cd fluent.inc → enter



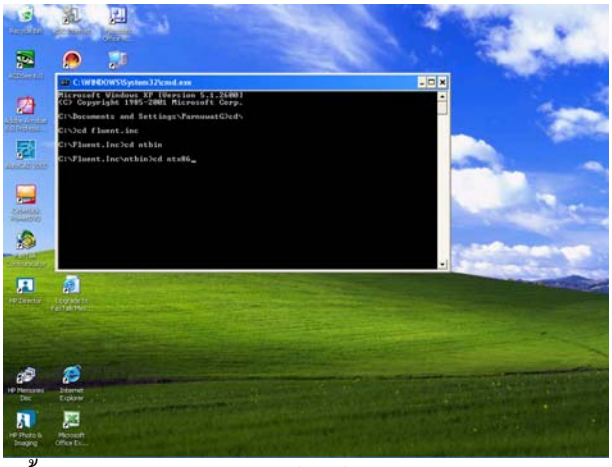
ภาพผนวกที่ ค7 แสดงขั้นตอนการใช้งานซอฟต์แวร์ Fluent แบบขนานบน Windows ขั้นที่ 4

5. พิมพ์ cd ntbin → enter



ภาพผนวกที่ ค8 แสดงขั้นตอนการใช้งานซอฟต์แวร์ Fluent แบบขนานบน Windows ขั้นที่ 5

6. พิมพ์ cd ntx86 → enter



ภาพผนวกที่ ค9 แสดงขั้นตอนการใช้งานซอฟต์แวร์ Fluent แบบขนานบน Windows ขั้นที่ 6

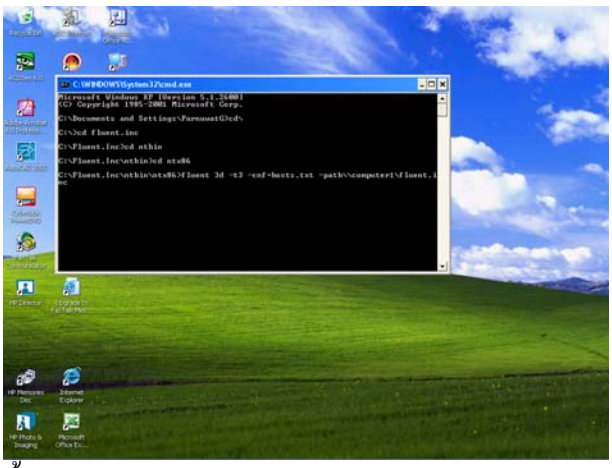
7. พิมพ์ Fluent 3d –t3 –cnf=host.txt –path\\computer1\fluent.inc

   3d หมายถึงมิติของงานที่จะนำมาคำนวณ

   -t3 หมายถึงจำนวนเครื่องที่ใช้คำนวณ (เครื่องแม่รวมกับเครื่องลูก)

   host.txt หมายถึงต้องกำหนดชื่อของเครื่องคอมพิวเตอร์ที่จะมาใช้คำนวน

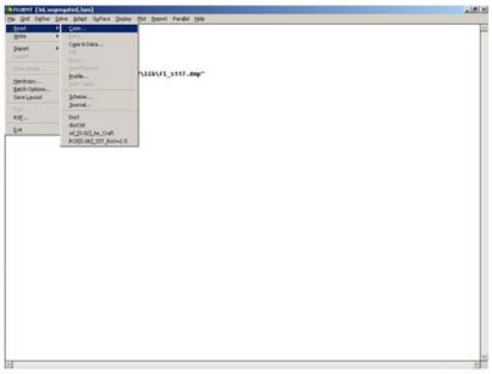
   computer1\fluent.inc หมายถึงกำหนดตำแหน่งของซอฟต์แวร์ Fluent ในเครื่องแม่



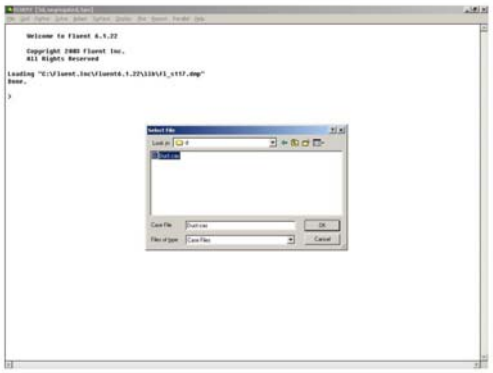
ภาพผนวกที่ ค10 แสดงขั้นตอนการใช้งานซอฟต์แวร์ Fluent แบบขนานบน Windows ขั้นที่ 7

ภาคผนวก ง คู่มือการใช้ฟังก์ชั่นยูดีเอฟบนระบบปฏิบัติการ Windows และ LINUX

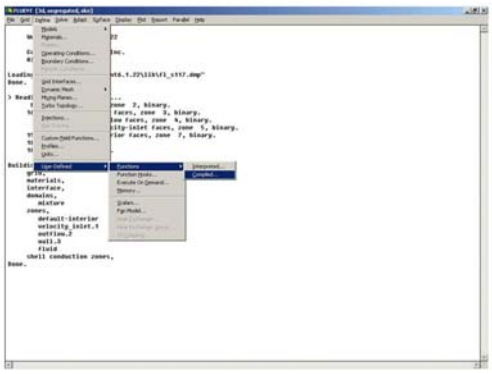
คู่มือการใช้งานฟังก์ชั่นยูดีเอฟ บนระบบปฏิบัติการ Windows และ LINUX



ภาพผนวกที่ ง1 แสดงขั้นตอนการใช้ฟังก์ชั่นยูดีเอฟขั้นที่ 1

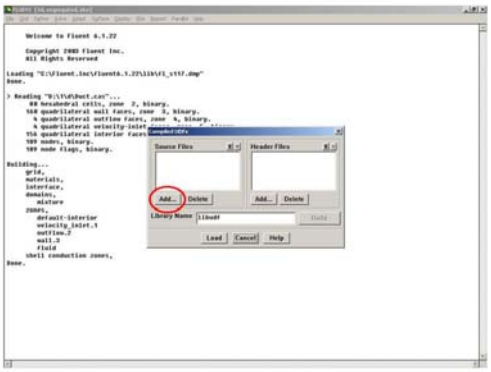


ภาพผนวกที่ ง2 แสดงขั้นตอนการใช้ฟังก์ชั่นยูดีเอฟขั้นที่ 2



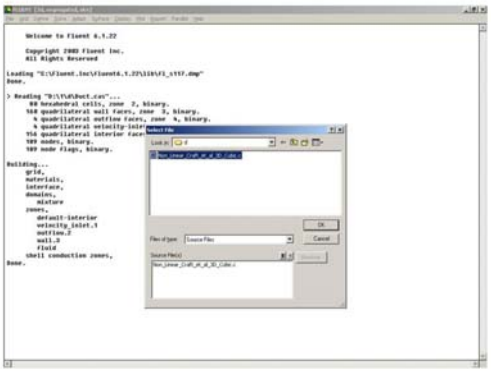
CML-ME-KU เลือก Define ---> User-defined ---> Functions ---> Compiled 4

ภาพผนวกที่ ง3 แสดงขั้นตอนการใช้ฟังก์ชั่นยูดีเอฟขั้นที่ 3



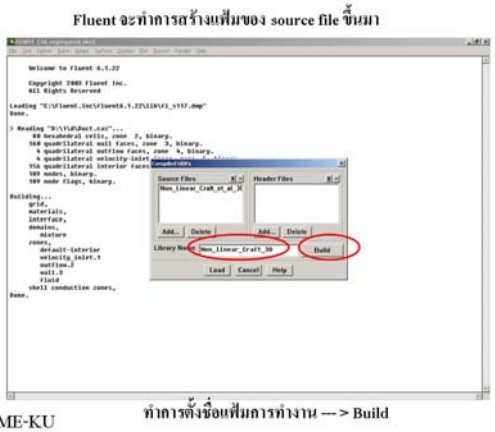
CML-ME-KU เลือก Source file ---> Add 5

ภาพผนวกที่ ง4 แสดงขั้นตอนการใช้ฟังก์ชั่นยูดีเอฟขั้นที่ 4



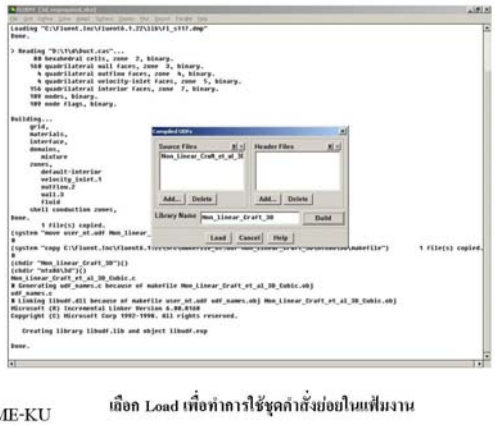
CML-ME-KU เลือก source file ที่ต้องการ ---> OK 6

ภาพผนวกที่ ง5 แสดงขั้นตอนการใช้ฟังก์ชั่นยูดีเอฟขั้นที่ 5

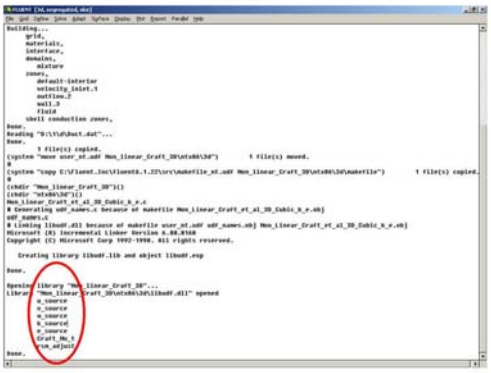


ภาพผนวกที่ ง6 แสดงขั้นตอนการใช้ฟังก์ชั่นยูดีเอฟขั้นที่ 6

ภาพผนวกที่ ง7 แสดงขั้นตอนการใช้ฟังก์ชั่นยูดีเอฟขั้นที่ 7



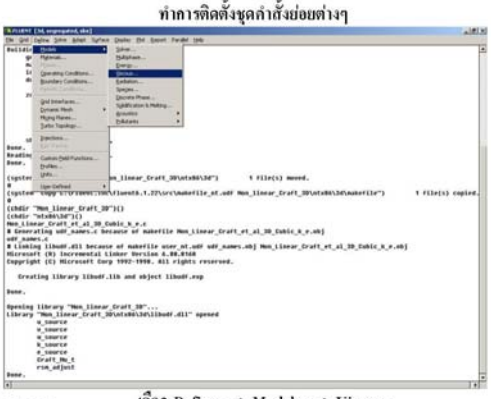
ภาพผนวกที่ ง8 แสดงขั้นตอนการใช้ฟังก์ชั่นยูดีเอฟขั้นที่ 8



CML-ME-KU Fluent แสดงชุดคำสั่งย่อยที่ถูกสร้างในแฟ้มงาน 10

ภาพผนวกที่ ง9 แสดงขั้นตอนการใช้ฟังก์ชั่นยูดีเอฟขั้นที่ 9

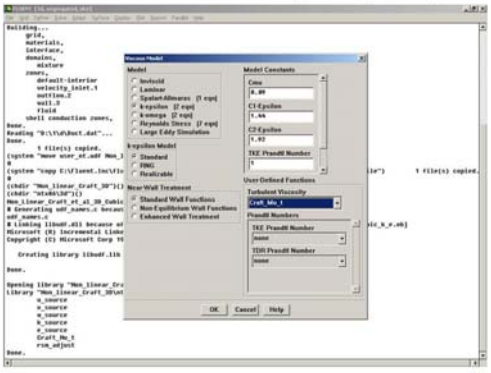
ทำการติดตั้งชุดคำสั่งย่อยต่างๆ



CML-ME-KU เลือก Define --- > Models --- > Viscous 11

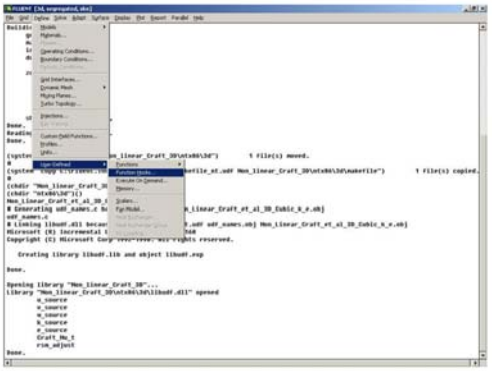
ภาพผนวกที่ ง10 แสดงขั้นตอนการใช้ฟังก์ชั่นยูดีเอฟขั้นที่ 10

ทำการเปลี่ยน Turbulent Viscosity หรือ Eddy Viscosity



CML-ME-KU เลือก Turbulent Viscosity --- > Craft_Mu_t 12

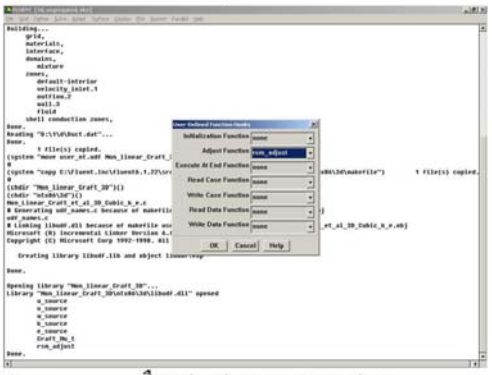
ภาพผนวกที่ ง11 แสดงขั้นตอนการใช้ฟังก์ชั่นยูดีเอฟขั้นที่ 11



CML-ME-KU เลือก Define ---> User-Defined ---> Function Hooks 13

ภาพผนวกที่ ง12 แสดงขั้นตอนการใช้ฟังก์ชั่นยูดีเอฟขั้นที่ 12

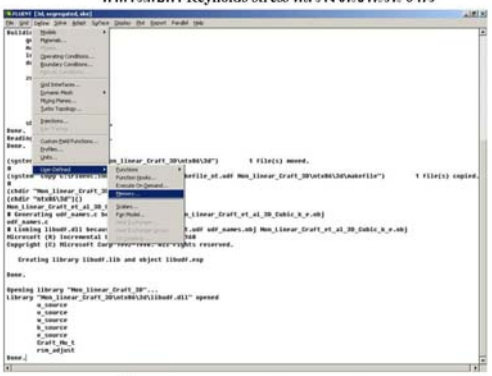
ทำการติดตั้งตัว Reynolds stress ที่สร้างขึ้นมา



CML-ME-KU เลือก Adjust function ---> rsm_adjust 14

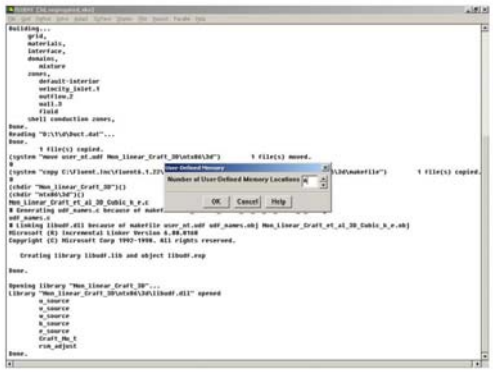
ภาพผนวกที่ ง13 แสดงขั้นตอนการใช้ฟังก์ชั่นยูดีเอฟขั้นที่ 13

ทำการเก็บค่า Reynolds stress ที่สร้างขึ้นจำนวน 6 ตัว



CML-ME-KU เลือก Define ---> User-Defined ---> Memory 15

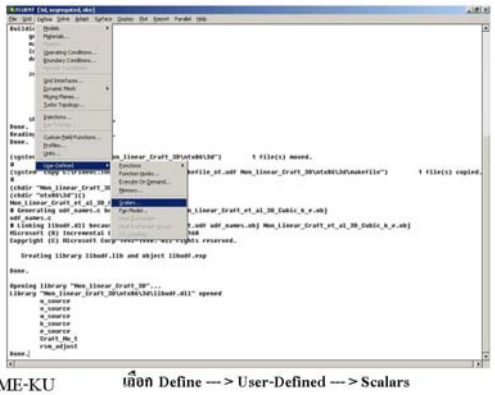
ภาพผนวกที่ ง14 แสดงขั้นตอนการใช้ฟังก์ชั่นยูดีเอฟขั้นที่ 14



CML-ME-KU เลือก 6 ---> OK 16

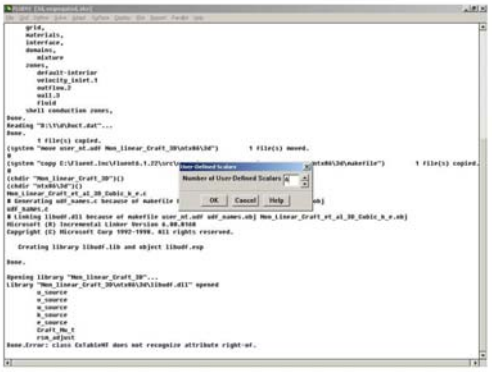
ภาพผนวกที่ ง15 แสดงขั้นตอนการใช้ฟังก์ชั่นยูดีเอฟขั้นที่ 15

ทำการกำหนดตัวแปร anisotropy tensor ที่สร้างจำนวน 6 ตัว



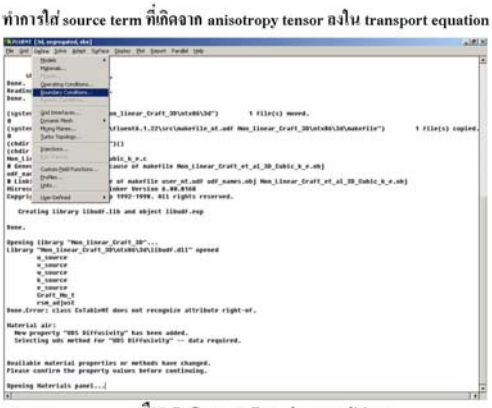
CML-ME-KU เลือก Define ---> User-Defined ---> Scalars 17

ภาพผนวกที่ ง16 แสดงขั้นตอนการใช้ฟังก์ชั่นยูดีเอฟขั้นที่ 16

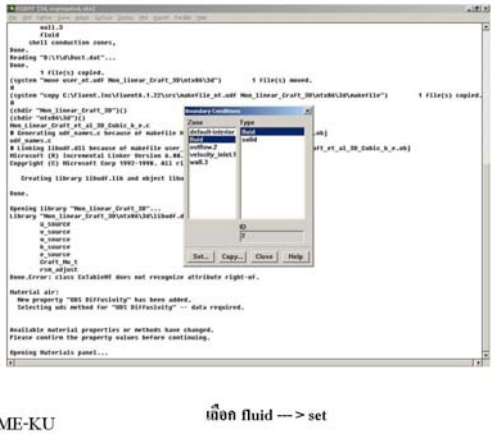


CML-ME-KU เลือก 6 ---> OK 18

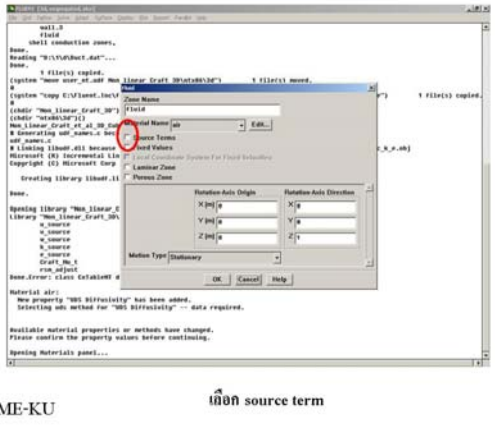
ภาพผนวกที่ ง17 แสดงขั้นตอนการใช้ฟังก์ชั่นยูดีเอฟขั้นที่ 17



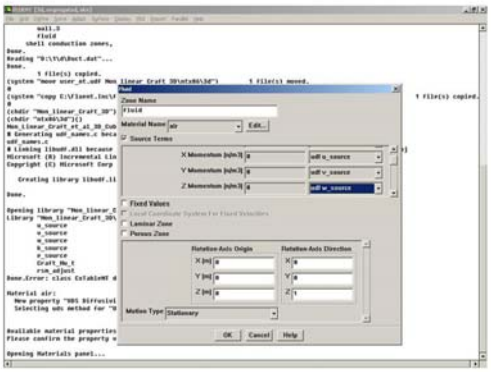
ภาพผนวกที่ ง18 แสดงขั้นตอนการใช้ฟังก์ชั่นยูดีเอฟขั้นที่ 18



ภาพผนวกที่ ง19 แสดงขั้นตอนการใช้ฟังก์ชั่นยูดีเอฟขั้นที่ 19

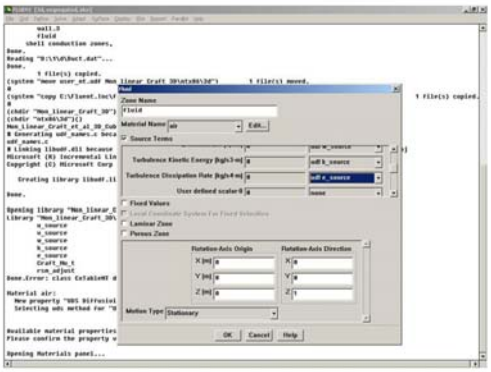


ภาพผนวกที่ ง20 แสดงขั้นตอนการใช้ฟังก์ชั่นยูดีเอฟขั้นที่ 20



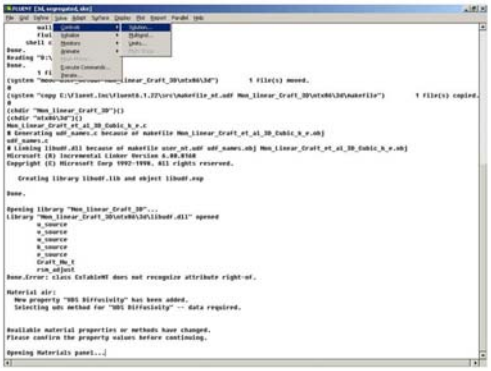
CML-ME-KU ทำการใส่ source term ลงใน transport equation 22

ภาพผนวกที่ ง21 แสดงขั้นตอนการใช้ฟังก์ชั่นยูดีเอฟขั้นที่ 21



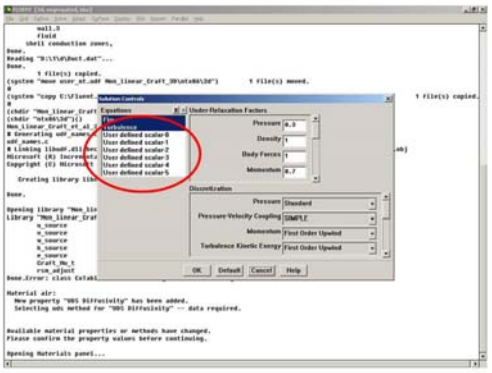
CML-ME-KU ทำการใส่ source term ลงใน transport equation ---> OK 23

ภาพผนวกที่ ง22 แสดงขั้นตอนการใช้ฟังก์ชั่นยูดีเอฟขั้นที่ 22



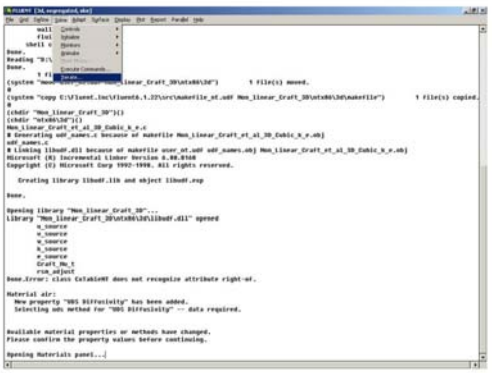
CML-ME-KU เลือก Solve ---> Controls ---> Solution 24

ภาพผนวกที่ ง23 แสดงขั้นตอนการใช้ฟังก์ชั่นยูดีเอฟขั้นที่ 23



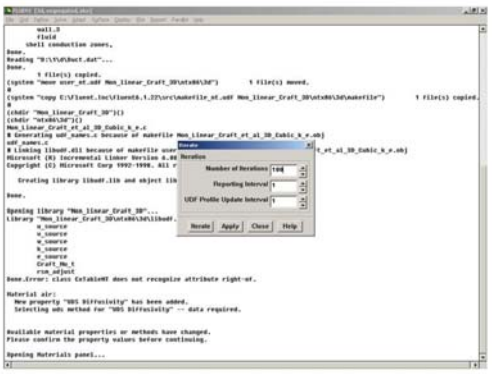
CML-ME-KU น้ำแดงดีของ User defined scalar ออก 25

ภาพผนวกที่ ง24 แสดงขั้นตอนการใช้ฟังก์ชั่นยูดีเอฟขั้นที่ 24



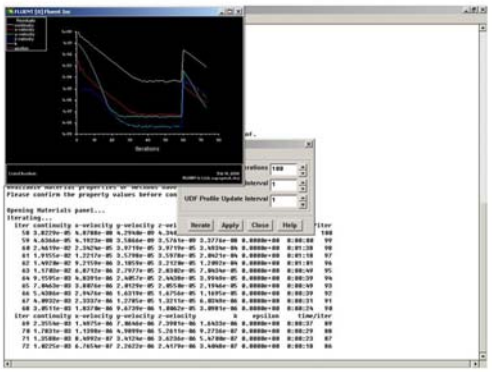
CML-ME-KU เลือก Solve ---> Iterate 26

ภาพผนวกที่ ง25 แสดงขั้นตอนการใช้ฟังก์ชั่นยูดีเอฟขั้นที่ 25

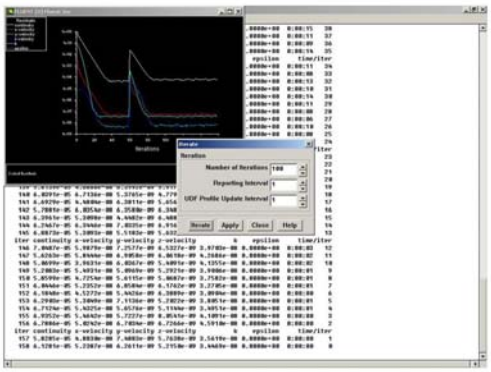


CML-ME-KU เลือก 100 ---> OK 27

ภาพผนวกที่ ง26 แสดงขั้นตอนการใช้ฟังก์ชั่นยูดีเอฟขั้นที่ 26



ภาพผนวกที่ ง27 แสดงขั้นตอนการใช้ฟังก์ชั่นยูดีเอฟขั้นที่ 27



ภาพผนวกที่ ง28 แสดงขั้นตอนการใช้ฟังก์ชั่นยูดีเอฟขั้นที่ 28

ภาคผนวก จ โปรแกรมของฟังก์ชันยูดีเอฟของแบบจำลองความปั่นป่วนชนิดไม่เชิงเส้น

โปรแกรมของฟังก์ชันยูดีเอฟสำหรับแบบจำลองความปั่นป่วนชนิดไม่เชิงเส้น แบบ low Reynolds number ของ Craft et al. (1996)

```
// Non Linear Craft et al 3D Cubic with k and epsilon source
//by Suabsakul Gururatana
//8 Feb 2006
#include "udf.h"
#include "math.h"
/* Turbulence model constants */
const real C_1=-0.1;
const real C_2=0.1;
const real C_3=0.26;
const real C_5=0.0;
// User-defined scalars : Define Reynolds stress
enum
{
UU,
VV,
WW,
UV,
UW,
VW
};
// Define source in x momentun equation
DEFINE_SOURCE(u_source, c, t, dS, eqn)
{
real source;
dS[eqn]=0.0;
source= - C_R(c,t) * ( C_UDSI_G(c,t,UU)[0] + C_UDSI_G(c,t,UV)[1]+C_UDSI_G(c,t,UW)[2] );
return source;
}
// Define source in y momentun equation
DEFINE_SOURCE(v_source, c, t, dS, eqn)
```

```
{
real source;
dS[eqn]= 0.0;
source = - C_R(c,t)*(C_UDSI_G(c,t,UV)[0]+ C_UDSI_G(c,t,VV)[1] + C_UDSI_G(c,t,VW)[2] ) ;
return source;
}
// Define source in z momentun equation
DEFINE_SOURCE(w_source, c, t, dS, eqn)
{
real source;
dS[eqn]= 0.0;
source= -C_R(c,t)*(C_UDSI_G(c,t,UW)[0] + C_UDSI_G(c,t,VW)[1] + C_UDSI_G(c,t,WW)[2] );
return source;
}
DEFINE_SOURCE(k_source, c, t, dS, eqn)
{
real prod;
dS[eqn]= 0.0;
prod=-C_R(c,t)*(C_UDSI(c,t,UU)*C_DUDX(c,t)+C_UDSI(c,t,UV)*C_DVDX(c,t)+
C_UDSI(c,t,UW)*C_DWDX(c,t)+C_UDSI(c,t,UV)*C_DUDY(c,t)+C_UDSI(c,t,VV)*C_DVDY(c,t)+C_UDSI(c,t,VW)*C_DWD
Y(c,t)+C_UDSI(c,t,UW)*C_DUDZ(c,t)+C_UDSI(c,t,VW)*C_DVDZ(c,t) + C_UDSI(c,t,WW)*C_DWDZ(c,t));
return prod;
}

DEFINE_SOURCE(e_source, c, t, dS, eqn)
{
real prod;
real c_1e =1.44;
real c_2e =1.92;
real k   = C_K(c,t);
real e   = C_D(c,t);
real u = C_MU_L(c,t);
real p   = C_R(c,t);
real v, Ret,fe2,cc;
                v = u/p;
                Ret=k*k/(e*v);
```

```
            dS[eqn]= 0.0;
            prod= - c_1e*C_D(c,t)/C_K(c,t)* C_R(c,t)
                                                    *       (       C_UDSI(c,t,UU)*C_DUDX(c,t)      +
C_UDSI(c,t,UV)*C_DVDX(c,t) + C_UDSI(c,t,UW)*C_DWDX(c,t)+
                                                            C_UDSI(c,t,UV)*C_DUDY(c,t)              +
C_UDSI(c,t,VV)*C_DVDY(c,t) + C_UDSI(c,t,VW)*C_DWDY(c,t)+
                                                            C_UDSI(c,t,UW)*C_DUDZ(c,t)              +
C_UDSI(c,t,VW)*C_DVDZ(c,t) + C_UDSI(c,t,WW)*C_DWDZ(c,t)
                                    );
            fe2=1.0- 0.3*(exp(-Ret*Ret));
            return prod- (c_2e*C_R(c,t)*e*e/k)+ (fe2*c_2e*C_R(c,t)*e*e/k );
}
// Define Eddy  viscosity
DEFINE_TURBULENT_VISCOSITY(Craft_Mu_t,c,t)
{
real S,W,c_mu,T,mu_t;
real S11, S12, S13, S21, S22, S23, S31, S32, S33;
real W11, W12, W13, W21, W22, W23, W31, W32, W33;
// Define variable
real u = C_MU_L(c,t); //laminar viscosity
real p   = C_R(c,t);     //density
real v, Ret,fmu;
real k   = C_K(c,t);     //Turbulent kinetic energy(k)
real e   = C_D(c,t);     //Dissipation rate (epsilon)
            T=C_K(c,t)/C_D(c,t); // k/e
// Define stress tensor
                                        S11 = 0.5*( C_DUDX(c,t)+C_DUDX(c,t) );
                                        S12 = 0.5*( C_DUDY(c,t)+C_DVDX(c,t) );
                                        S13 = 0.5*( C_DUDZ(c,t)+C_DWDX(c,t) );
                                        S21 = 0.5*( C_DVDX(c,t)+C_DUDY(c,t) );
                                        S22 = 0.5*( C_DVDY(c,t)+C_DVDY(c,t) );
                                        S23 = 0.5*( C_DVDZ(c,t)+C_DWDY(c,t) );
                                        S31 = 0.5*( C_DWDX(c,t)+C_DUDZ(c,t) );
                                        S32 = 0.5*( C_DWDY(c,t)+C_DVDZ(c,t) );
                                        S33 = 0.5*( C_DWDZ(c,t)+C_DWDZ(c,t) );
S=T*sqrt( 2*(S11*S11+S12*S12+S13*S13+S21*S21+S22*S22+S23*S23+S31*S31+S32*S32+S33*S33) );
```

```
// Define vorticity tensor
                                        W11 = 0.5*( C_DUDX(c,t)-C_DUDX(c,t) );
                                        W12 = 0.5*( C_DUDY(c,t)-C_DVDX(c,t) );
                                        W13 = 0.5*( C_DUDZ(c,t)-C_DWDX(c,t) );
                                        W21 = 0.5*( C_DVDX(c,t)-C_DUDY(c,t) );
                                        W22 = 0.5*( C_DVDY(c,t)-C_DVDY(c,t) );
                                        W23 = 0.5*( C_DVDZ(c,t)-C_DWDY(c,t) );
                                        W31 = 0.5*( C_DWDX(c,t)-C_DUDZ(c,t) );
                                        W32 = 0.5*( C_DWDY(c,t)-C_DVDZ(c,t) );
                                        W33 = 0.5*( C_DWDZ(c,t)-C_DWDZ(c,t) );
W=T*sqrt( 2*(W11*W11+W12*W12+W13*W13+W21*W21+W22*W22+W23*W23+W31*W31+W32*W32+W33*W33) );
// Define kinenatic viscosity
v = u/p;
// Define turbulent Reynolds number
Ret=k*k/(e*v);
// Define Constant Cmu
c_mu=(0.3/(1.0+(0.35* MAX(S,W) )))*(1.0-exp(-0.36/(exp(-0.75* MAX(S,W) ))));
// Define Damping function of Craft et al.(1996)
fmu=(1.0-exp(-pow((Ret/90.),0.5)-pow((Ret/400.),2) ));
// Define Eddy viscosity
mu_t=c_mu*T*C_K(c,t);
return mu_t;
}
DEFINE_ADJUST(rsm_adjust,domain)
{
Thread  *t;
cell_t  c;
real X;
real mu_t, c_mu;
real S11, S12, S13, S21, S22, S23, S31, S32, S33;
real W11, W12, W13, W21, W22, W23, W31, W32, W33;
real S, W, Sij, Wij;
real C_4, C_6, C_7;
real P11, P22, P33, Pkk;
real tau_w, u_tauw, u_star, y_star, Gk, u_mag;
real u ;
```

```
real p ;
real v, Ret,fmu;
real k ;
real e ;
/* Set the turbulent viscosity */
thread_loop_c(t,domain)
if (FLUID_THREAD_P(t))
{
begin_c_loop(c,t)
{
// Define variable
u= C_MU_L(c,t); //laminar viscosity
p= C_R(c,t);            //density
k= C_K(c,t);            //Turbulent kinetic energy(k)
e= C_D(c,t);            //Dissipation rate (epsilon)
X=C_K(c,t)/C_D(c,t); // k/e
// Define stress tensor
                                        S11 = 0.5*( C_DUDX(c,t)+C_DUDX(c,t) );
                                        S12 = 0.5*( C_DUDY(c,t)+C_DVDX(c,t) );
                                        S13 = 0.5*( C_DUDZ(c,t)+C_DWDX(c,t) );
                                        S21 = 0.5*( C_DVDX(c,t)+C_DUDY(c,t) );
                                        S22 = 0.5*( C_DVDY(c,t)+C_DVDY(c,t) );
                                        S23 = 0.5*( C_DVDZ(c,t)+C_DWDY(c,t) );
                                        S31 = 0.5*( C_DWDX(c,t)+C_DUDZ(c,t) );
                                        S32 = 0.5*( C_DWDY(c,t)+C_DVDZ(c,t) );
                                        S33 = 0.5*( C_DWDZ(c,t)+C_DWDZ(c,t) );
S=X*sqrt( 2*(S11*S11+S12*S12+S13*S13+S21*S21+S22*S22+S23*S23+S31*S31+S32*S32+S33*S33) );
// Define vorticity tensor
                                        W11 = 0.5*( C_DUDX(c,t)-C_DUDX(c,t) );
                                        W12 = 0.5*( C_DUDY(c,t)-C_DVDX(c,t) );
                                        W13 = 0.5*( C_DUDZ(c,t)-C_DWDX(c,t) );
                                        W21 = 0.5*( C_DVDX(c,t)-C_DUDY(c,t) );
                                        W22 = 0.5*( C_DVDY(c,t)-C_DVDY(c,t) );
                                        W23 = 0.5*( C_DVDZ(c,t)-C_DWDY(c,t) );
                                        W31 = 0.5*( C_DWDX(c,t)-C_DUDZ(c,t) );
                                        W32 = 0.5*( C_DWDY(c,t)-C_DVDZ(c,t) );
```

```
                                        W33 = 0.5*( C_DWDZ(c,t)-C_DWDZ(c,t) );
W=X*sqrt( 2*(W11*W11+W12*W12+W13*W13+W21*W21+W22*W22+W23*W23+W31*W31+W32*W32+W33*W33) );
// Define kinenatic viscosity
v = u/p;
// Define turbulent Reynolds number
Ret=k*k/(e*v);
// Define Constant Cmu
c_mu=(0.3/(1.0+(0.35* MAX(S,W) )))*(1.0-exp(-0.36/(exp(-0.75* MAX(S,W) ))));
// Define Damping function of Craft et al.(1996)
fmu=(1.0-exp(-pow((Ret/90.),0.5)-pow((Ret/400.),2) ));
// Define Eddy viscosity
mu_t=c_mu*fmu*C_K(c,t)*C_K(c,t)*C_D(c,t);
// Define constant of Craft et al.(1996)
C_4=-10.0*SQR(c_mu);
C_6=-5.0*SQR(c_mu);
C_7=5.0*SQR(c_mu);
// Define tensor index = SklSkl
Sij=S11*S11+S12*S12+S13*S13+S21*S21+S22*S22+S23*S23+S31*S31+S32*S32+S33*S33;
// Define tensor index = WklWkl
Wij=W11*W11+W12*W12+W13*W13+W21*W21+W22*W22+W23*W23+W31*W31+W32*W32+W33*W33;
//*************************************************************************//
//************** IMPORTANT define only anisotropy tensor ******************//
//************** and multiply k in the Reynolds stress   ******************//
//**************  anisotropy tensor*****************************************//
//*************************************************************************//
//*************************************************************************//
// Define anisotropy tensor of normal Reynolds stress
C_UDSI(c,t,UU)=
                                         C_1*mu_t*X*   (S11*S11+S12*S12+S13*S13-1./3.*Sij)
                                        + 2.*C_2*mu_t*X*   (W12*S12+W13*S13)
                                        + C_3*mu_t*X*   (W12*W12+W13*W13-1./3.*Wij)
                                        -2.*C_4*mu_t*X*X*
(W12*(S11*S12+S12*S22+S13*S23)+W13*(S11*S13+S12*S23+S13*S33))
                                        +    C_6*mu_t*X*X* (S11*Sij)
                                        +    C_7*mu_t*X*X* (S11*Wij);
```

```
C_UDSI(c,t,VV)=
                                        C_1*mu_t*X*   (S12*S12+S22*S22+S23*S23-1./3.*Sij)
                                      + 2.*C_2*mu_t*X*   (W23*S23-W12*S12)
                                       +   C_3*mu_t*X*   (W12*W12+W23*W23-1./3.*Wij)
                                      +              2.*C_4*mu_t*X*X*              (W12*(S11*S12+S12*S22+S13*S23)-
W23*(S12*S13+S22*S23+S23*S33))
                                       +   C_6*mu_t*X*X* (S22*Sij)
                                       +   C_7*mu_t*X*X* (S22*Wij);


C_UDSI(c,t,WW)=
                                        C_1*mu_t*X*   (S13*S13+S23*S23+S33*S33-1./3.*Sij)
                                      - 2.*C_2*mu_t*X*   (W13*S13+W23*S23)
                                      + C_3*mu_t*X*   (W13*W13+W23*W23-1./3.*Wij)
                                      +2.*C_4*mu_t*X*X*
(W13*(S11*S13+S12*S23+S13*S33)+W23*(S12*S13+S22*S23+S23*S33))
                                      +   C_6*mu_t*X*X* (S33*Sij)
                                      +   C_7*mu_t*X*X* (S33*Wij);
// Define anisotropy tensor of shear Reynolds stress
        C_UDSI(c,t,UV)=
                                        C_1*mu_t*X*   (S11*S12+S12*S22+S13*S23)
                                      +   C_2*mu_t*X*   (W12*(S22-S11)+W13*S23+W23*S13)
                                      +   C_3*mu_t*X*   (W13*W23)
                                      +          C_4*mu_t*X*X*          (W12*(S11*S11+S13*S13-S22*S22-S23*S23)-
W13*(S12*S13+S22*S23+S23*S33)-W23*(S11*S13+S12*S23+S13*S33))
                                      +   C_6*mu_t*X*X* (S12*Sij)
                                      +   C_7*mu_t*X*X* (S12*Wij);


        C_UDSI(c,t,UW)=
                                        C_1*mu_t*X*   (S11*S13+S12*S23+S13*S33)
                                      +   C_2*mu_t*X*   (W13*(S33-S11)+W12*S23-W23*S12)
                                      -   C_3*mu_t*X*   (W12*W23)
                                      +                C_4*mu_t*X*X*     (W13*(S11*S11+S12*S12-S23*S23-S33*S33)-
W12*(S11*S13+S22*S23+S23*S33)+W23*(S11*S12+S12*S22+S13*S23))
                                      +   C_6*mu_t*X*X* (S13*Sij)
                                      +   C_7*mu_t*X*X* (S13*Wij);
```

```
C_UDSI(c,t,VW)=
                                        C_1*mu_t*X*   (S12*S13+S22*S23+S23*S33)
                                    +   C_2*mu_t*X*   (W23*(S33-S22)-W12*S13-W13*S12)
                                    +   C_3*mu_t*X*   (W12*W13)
                                    +                 C_4*mu_t*X*X*       (W23*(S12*S12+S22*S22-S13*S13-
S33*S33)+W12*(S11*S13+S12*S23+S13*S33)+W13*(S11*S12+S12*S22+S13*S23))
                                    +   C_6*mu_t*X*X* (S23*Sij)
                                    +   C_7*mu_t*X*X* (S23*Wij);
// Define memory Reynolds stress by User defined Memory
//*********************************************************************//
//*************IMPORTANT define Linear and Nonlinear Term*****************//
//*********************************************************************//
C_UDMI(c,t,0)=        2./3.*C_K(c,t) -2.*mu_t*S11  + C_UDSI(c,t,UU);
C_UDMI(c,t,1)=        2./3.*C_K(c,t) -2.*mu_t*S22  + C_UDSI(c,t,VV);
C_UDMI(c,t,2)=        2./3.*C_K(c,t) -2.*mu_t*S33  + C_UDSI(c,t,WW);
C_UDMI(c,t,3)=       -2.*mu_t*S12  + C_UDSI(c,t,UV);
C_UDMI(c,t,4)=       -2.*mu_t*S13  + C_UDSI(c,t,UW);
C_UDMI(c,t,5)=       -2.*mu_t*S23  + C_UDSI(c,t,VW);
}
end_c_loop(c,t)
}
}
```

โปรแกรมของฟังก์ชันยูดีเอฟสำหรับแบบจำลองความปั่นป่วนชนิดไม่เชิงเส้นแบบ low Reynolds number ของ Craft et al. (1996) ที่ใช้ damping function ของ Gibson and Dafa'Alla (1994)

```
// Non Linear Craft et al 3D Cubic +Fmu of Gibson and Dafa'Alla (1994) +k and e source
//by Suabsakul Gururatana
//8 Feb 2006
#include "udf.h"
#include "math.h"
/* Turbulence model constants */
const real C_1=-0.1;
const real C_2=0.1;
const real C_3=0.26;
```

```
const real C_5=0.0;
// User-defined scalars : Define Reynolds stress
enum
{
UU,
VV,
WW,
UV,
UW,
VW
};
// Define source in x momentun equation
DEFINE_SOURCE(u_source, c, t, dS, eqn)
{
real source;
dS[eqn]=0.0;
source= - C_R(c,t) * ( C_UDSI_G(c,t,UU)[0] + C_UDSI_G(c,t,UV)[1]+C_UDSI_G(c,t,UW)[2] );
return source;
}
// Define source in y momentun equation
DEFINE_SOURCE(v_source, c, t, dS, eqn)
{
real source;
dS[eqn]= 0.0;
source = - C_R(c,t)*(C_UDSI_G(c,t,UV)[0]+ C_UDSI_G(c,t,VV)[1] + C_UDSI_G(c,t,VW)[2] ) ;
return source;
}
// Define source in z momentun equation
DEFINE_SOURCE(w_source, c, t, dS, eqn)
{
real source;
dS[eqn]= 0.0;
source= -C_R(c,t)*(C_UDSI_G(c,t,UW)[0] + C_UDSI_G(c,t,VW)[1] + C_UDSI_G(c,t,WW)[2] );
return source;
}
DEFINE_SOURCE(k_source, c, t, dS, eqn)
```

```
{
real prod;
dS[eqn]= 0.0;
prod=-C_R(c,t)*(C_UDSI(c,t,UU)*C_DUDX(c,t)+C_UDSI(c,t,UV)*C_DVDX(c,t)+
C_UDSI(c,t,UW)*C_DWDX(c,t)+C_UDSI(c,t,UV)*C_DUDY(c,t)+C_UDSI(c,t,VV)*C_DVDY(c,t)+C_UDSI(c,t,VW)*C_DWD
Y(c,t)+C_UDSI(c,t,UW)*C_DUDZ(c,t)+C_UDSI(c,t,VW)*C_DVDZ(c,t) + C_UDSI(c,t,WW)*C_DWDZ(c,t));
return prod;
}
DEFINE_SOURCE(e_source, c, t, dS, eqn)
{
real prod;
real c_1e =1.44;
real c_2e =1.92;
real k   = C_K(c,t);
real e   = C_D(c,t);
real u = C_MU_L(c,t);
real p   = C_R(c,t);
real v, Ret,fe2,cc;
        v = u/p;
        Ret=k*k/(e*v);
        dS[eqn]= 0.0;
        prod= - c_1e*C_D(c,t)/C_K(c,t)* C_R(c,t)
                                                * ( C_UDSI(c,t,UU)*C_DUDX(c,t) +
C_UDSI(c,t,UV)*C_DVDX(c,t) + C_UDSI(c,t,UW)*C_DWDX(c,t)+
                                                  C_UDSI(c,t,UV)*C_DUDY(c,t) +
C_UDSI(c,t,VV)*C_DVDY(c,t) + C_UDSI(c,t,VW)*C_DWDY(c,t)+
                                                  C_UDSI(c,t,UW)*C_DUDZ(c,t) +
C_UDSI(c,t,VW)*C_DVDZ(c,t) + C_UDSI(c,t,WW)*C_DWDZ(c,t)
                                    );
        fe2=1.0- 0.3*(exp(-Ret*Ret));
        return prod- (c_2e*C_R(c,t)*e*e/k)+ (fe2*c_2e*C_R(c,t)*e*e/k );
}

// Define Eddy  viscosity
DEFINE_TURBULENT_VISCOSITY(Craft_Mu_t,c,t)
{
```

```
real S,W,c_mu,T,mu_t;
real S11, S12, S13, S21, S22, S23, S31, S32, S33;
real W11, W12, W13, W21, W22, W23, W31, W32, W33;
// Define variable
real u = C_MU_L(c,t); //laminar viscosity
real p   = C_R(c,t);      //density
real v, Ret,fmu;
real k   = C_K(c,t);      //Turbulent kinetic energy(k)
real e   = C_D(c,t);      //Dissipation rate (epsilon)
              T=C_K(c,t)/C_D(c,t); // k/e
// Define stress tensor
                                                S11 = 0.5*( C_DUDX(c,t)+C_DUDX(c,t) );
                                                S12 = 0.5*( C_DUDY(c,t)+C_DVDX(c,t) );
                                                S13 = 0.5*( C_DUDZ(c,t)+C_DWDX(c,t) );
                                                S21 = 0.5*( C_DVDX(c,t)+C_DUDY(c,t) );
                                                S22 = 0.5*( C_DVDY(c,t)+C_DVDY(c,t) );
                                                S23 = 0.5*( C_DVDZ(c,t)+C_DWDY(c,t) );
                                                S31 = 0.5*( C_DWDX(c,t)+C_DUDZ(c,t) );
                                                S32 = 0.5*( C_DWDY(c,t)+C_DVDZ(c,t) );
                                                S33 = 0.5*( C_DWDZ(c,t)+C_DWDZ(c,t) );
S=T*sqrt( 2*(S11*S11+S12*S12+S13*S13+S21*S21+S22*S22+S23*S23+S31*S31+S32*S32+S33*S33) );
// Define vorticity tensor
                                                W11 = 0.5*( C_DUDX(c,t)-C_DUDX(c,t) );
                                                W12 = 0.5*( C_DUDY(c,t)-C_DVDX(c,t) );
                                                W13 = 0.5*( C_DUDZ(c,t)-C_DWDX(c,t) );
                                                W21 = 0.5*( C_DVDX(c,t)-C_DUDY(c,t) );
                                                W22 = 0.5*( C_DVDY(c,t)-C_DVDY(c,t) );
                                                W23 = 0.5*( C_DVDZ(c,t)-C_DWDY(c,t) );
                                                W31 = 0.5*( C_DWDX(c,t)-C_DUDZ(c,t) );
                                                W32 = 0.5*( C_DWDY(c,t)-C_DVDZ(c,t) );
                                                W33 = 0.5*( C_DWDZ(c,t)-C_DWDZ(c,t) );
W=T*sqrt( 2*(W11*W11+W12*W12+W13*W13+W21*W21+W22*W22+W23*W23+W31*W31+W32*W32+W33*W33) );
// Define kinenatic viscosity
v = u/p;
// Define turbulent Reynolds number
Ret=k*k/(e*v);
```

```
// Define Constant Cmu
c_mu=(0.3/(1.0+(0.35* MAX(S,W) )))*(1.0-exp(-0.36/(exp(-0.75* MAX(S,W) ))));
// Define Damping function of Gibson and Dafa'Alla (1994)
fmu= exp(-6.0/((1.0+(Ret/50.0))*(1.0+(Ret/50.0))))*(1.0+(3.0*exp(-Ret/10.0)));
// Define Eddy viscosity
mu_t=c_mu*T*C_K(c,t);
return mu_t;
}
DEFINE_ADJUST(rsm_adjust,domain)
{
Thread  *t;
cell_t  c;
real X;
real mu_t, c_mu;
real S11, S12, S13, S21, S22, S23, S31, S32, S33;
real W11, W12, W13, W21, W22, W23, W31, W32, W33;
real S, W, Sij, Wij;
real C_4, C_6, C_7;
real P11, P22, P33, Pkk;
real tau_w, u_tauw, u_star, y_star, Gk, u_mag;
real u ;
real p ;
real v, Ret,fmu;
real k ;
real e ;
/* Set the turbulent viscosity */
thread_loop_c(t,domain)
if (FLUID_THREAD_P(t))
{
begin_c_loop(c,t)
{
// Define variable
u= C_MU_L(c,t); //laminar viscosity
p= C_R(c,t);              //density
k= C_K(c,t);              //Turbulent kinetic energy(k)
e= C_D(c,t);              //Dissipation rate (epsilon)
```

```
X=C_K(c,t)/C_D(c,t); // k/e

// Define stress tensor

                                        S11 = 0.5*( C_DUDX(c,t)+C_DUDX(c,t) );

                                        S12 = 0.5*( C_DUDY(c,t)+C_DVDX(c,t) );

                                        S13 = 0.5*( C_DUDZ(c,t)+C_DWDX(c,t) );

                                        S21 = 0.5*( C_DVDX(c,t)+C_DUDY(c,t) );

                                        S22 = 0.5*( C_DVDY(c,t)+C_DVDY(c,t) );

                                        S23 = 0.5*( C_DVDZ(c,t)+C_DWDY(c,t) );

                                        S31 = 0.5*( C_DWDX(c,t)+C_DUDZ(c,t) );

                                        S32 = 0.5*( C_DWDY(c,t)+C_DVDZ(c,t) );

                                        S33 = 0.5*( C_DWDZ(c,t)+C_DWDZ(c,t) );

S=X*sqrt( 2*(S11*S11+S12*S12+S13*S13+S21*S21+S22*S22+S23*S23+S31*S31+S32*S32+S33*S33) );

// Define vorticity tensor

                                        W11 = 0.5*( C_DUDX(c,t)-C_DUDX(c,t) );

                                        W12 = 0.5*( C_DUDY(c,t)-C_DVDX(c,t) );

                                        W13 = 0.5*( C_DUDZ(c,t)-C_DWDX(c,t) );

                                        W21 = 0.5*( C_DVDX(c,t)-C_DUDY(c,t) );

                                        W22 = 0.5*( C_DVDY(c,t)-C_DVDY(c,t) );

                                        W23 = 0.5*( C_DVDZ(c,t)-C_DWDY(c,t) );

                                        W31 = 0.5*( C_DWDX(c,t)-C_DUDZ(c,t) );

                                        W32 = 0.5*( C_DWDY(c,t)-C_DVDZ(c,t) );

                                        W33 = 0.5*( C_DWDZ(c,t)-C_DWDZ(c,t) );

W=X*sqrt( 2*(W11*W11+W12*W12+W13*W13+W21*W21+W22*W22+W23*W23+W31*W31+W32*W32+W33*W33) );

// Define kinenatic viscosity

v = u/p;

// Define turbulent Reynolds number

Ret=k*k/(e*v);

// Define Constant Cmu

c_mu=(0.3/(1.0+(0.35* MAX(S,W) )))*(1.0-exp(-0.36/(exp(-0.75* MAX(S,W) ))));

// Define Damping function of Gibson and Dafa'Alla (1994)

fmu= exp(-6.0/((1.0+(Ret/50.0))*(1.0+(Ret/50.0))))*(1.0+(3.0*exp(-Ret/10.0)));

// Define Eddy viscosity

mu_t=c_mu*fmu*C_K(c,t)*C_K(c,t)*C_D(c,t);

// Define constant of Craft et al.(1996)

C_4=-10.0*SQR(c_mu);

C_6=-5.0*SQR(c_mu);
```

```
C_7=5.0*SQR(c_mu);
// Define tensor index = SklSkl
Sij=S11*S11+S12*S12+S13*S13+S21*S21+S22*S22+S23*S23+S31*S31+S32*S32+S33*S33;
// Define tensor index = WklWkl
Wij=W11*W11+W12*W12+W13*W13+W21*W21+W22*W22+W23*W23+W31*W31+W32*W32+W33*W33;
//*********************************************************************//
//************** IMPORTANT define only anisotropy tensor ******************//
//************** and multiply k in the Reynolds stress  ************************//
//**************  anisotropy tensor*****************************************//
//*********************************************************************//
//*********************************************************************//
// Define anisotropy tensor of normal Reynolds stress
C_UDSI(c,t,UU)=
                                        C_1*mu_t*X*  (S11*S11+S12*S12+S13*S13-1./3.*Sij)
                                       + 2.*C_2*mu_t*X*  (W12*S12+W13*S13)
                                       + C_3*mu_t*X*  (W12*W12+W13*W13-1./3.*Wij)
                                       -2.*C_4*mu_t*X*X*
(W12*(S11*S12+S12*S22+S13*S23)+W13*(S11*S13+S12*S23+S13*S33))
                                       +   C_6*mu_t*X*X* (S11*Sij)
                                       +   C_7*mu_t*X*X* (S11*Wij);

C_UDSI(c,t,VV)=
                                           C_1*mu_t*X*  (S12*S12+S22*S22+S23*S23-1./3.*Sij)
                                       + 2.*C_2*mu_t*X*  (W23*S23-W12*S12)
                                        +   C_3*mu_t*X*  (W12*W12+W23*W23-1./3.*Wij)
                                       +              2.*C_4*mu_t*X*X*              (W12*(S11*S12+S12*S22+S13*S23)-
W23*(S12*S13+S22*S23+S23*S33))
                                        +   C_6*mu_t*X*X* (S22*Sij)
                                        +   C_7*mu_t*X*X* (S22*Wij);

C_UDSI(c,t,WW)=
                                          C_1*mu_t*X*  (S13*S13+S23*S23+S33*S33-1./3.*Sij)
                                       - 2.*C_2*mu_t*X*  (W13*S13+W23*S23)
                                       + C_3*mu_t*X*  (W13*W13+W23*W23-1./3.*Wij)
                                       +2.*C_4*mu_t*X*X*
(W13*(S11*S13+S12*S23+S13*S33)+W23*(S12*S13+S22*S23+S23*S33))
```

```
                                        +   C_6*mu_t*X*X* (S33*Sij)
                                        +   C_7*mu_t*X*X* (S33*Wij);
// Define anisotropy tensor of shear Reynolds stress
        C_UDSI(c,t,UV)=
                                            C_1*mu_t*X*   (S11*S12+S12*S22+S13*S23)
                                        +   C_2*mu_t*X*   (W12*(S22-S11)+W13*S23+W23*S13)
                                        +   C_3*mu_t*X*   (W13*W23)
                                        +       C_4*mu_t*X*X*       (W12*(S11*S11+S13*S13-S22*S22-S23*S23)-
W13*(S12*S13+S22*S23+S23*S33)-W23*(S11*S13+S12*S23+S13*S33))
                                        +   C_6*mu_t*X*X* (S12*Sij)
                                        +   C_7*mu_t*X*X* (S12*Wij);

        C_UDSI(c,t,UW)=
                                            C_1*mu_t*X*   (S11*S13+S12*S23+S13*S33)
                                        +   C_2*mu_t*X*   (W13*(S33-S11)+W12*S23-W23*S12)
                                        -   C_3*mu_t*X*   (W12*W23)
                                        +           C_4*mu_t*X*X*     (W13*(S11*S11+S12*S12-S23*S23-S33*S33)-
W12*(S11*S13+S22*S23+S23*S33)+W23*(S11*S12+S12*S22+S13*S23))
                                        +   C_6*mu_t*X*X* (S13*Sij)
                                        +   C_7*mu_t*X*X* (S13*Wij);

        C_UDSI(c,t,VW)=
                                            C_1*mu_t*X*   (S12*S13+S22*S23+S23*S33)
                                        +   C_2*mu_t*X*   (W23*(S33-S22)-W12*S13-W13*S12)
                                        +   C_3*mu_t*X*   (W12*W13)
                                        +               C_4*mu_t*X*X*       (W23*(S12*S12+S22*S22-S13*S13-
S33*S33)+W12*(S11*S13+S12*S23+S13*S33)+W13*(S11*S12+S12*S22+S13*S23))
                                        +   C_6*mu_t*X*X* (S23*Sij)
                                        +   C_7*mu_t*X*X* (S23*Wij);
// Define memory Reynolds stress by User defined Memory
//**********************************************************************//
//*************IMPORTANT define Linear and Nonlinear Term*****************//
//**********************************************************************//
C_UDMI(c,t,0)=        2./3.*C_K(c,t) -2.*mu_t*S11  + C_UDSI(c,t,UU);
C_UDMI(c,t,1)=        2./3.*C_K(c,t) -2.*mu_t*S22  + C_UDSI(c,t,VV);
C_UDMI(c,t,2)=        2./3.*C_K(c,t) -2.*mu_t*S33  + C_UDSI(c,t,WW);
```

```
C_UDMI(c,t,3)=        -2.*mu_t*S12  + C_UDSI(c,t,UV);
C_UDMI(c,t,4)=        -2.*mu_t*S13  + C_UDSI(c,t,UW);
C_UDMI(c,t,5)=        -2.*mu_t*S23  + C_UDSI(c,t,VW);
}
end_c_loop(c,t)
}
}
```

โปรแกรมของฟังก์ชันยูดีเอฟสำหรับแบบจำลองความปั่นป่วนชนิดไม่เชิงเส้น แบบ high Reynolds number ของ Craft et al. (1996)

```
// High Reynolds number Non Linear Craft et al 3D Cubic with k and epsilon source for EWF //by Suabsakul Gururatana
//8 Feb 2006
#include "udf.h"
#include "math.h"
/* Turbulence model constants */
const real C_1=-0.1;
const real C_2=0.1;
const real C_3=0.26;
const real C_5=0.0;
// User-defined scalars : Define Reynolds stress
enum
{
UU,
VV,
WW,
UV,
UW,
VW
};
// Define source in x momentun equation
DEFINE_SOURCE(u_source, c, t, dS, eqn)
{
real source;
dS[eqn]=0.0;
```

```
source= - C_R(c,t) * ( C_UDSI_G(c,t,UU)[0] + C_UDSI_G(c,t,UV)[1]+C_UDSI_G(c,t,UW)[2] );
return source;
}
// Define source in y momentun equation
DEFINE_SOURCE(v_source, c, t, dS, eqn)
{
real source;
dS[eqn]= 0.0;
source = - C_R(c,t)*(C_UDSI_G(c,t,UV)[0]+ C_UDSI_G(c,t,VV)[1] + C_UDSI_G(c,t,VW)[2] ) ;
return source;
}
// Define source in z momentun equation
DEFINE_SOURCE(w_source, c, t, dS, eqn)
{
real source;
dS[eqn]= 0.0;
source= -C_R(c,t)*(C_UDSI_G(c,t,UW)[0] + C_UDSI_G(c,t,VW)[1] + C_UDSI_G(c,t,WW)[2] );
return source;
}
DEFINE_SOURCE(k_source, c, t, dS, eqn)
{
real prod;
dS[eqn]= 0.0;
prod=-C_R(c,t)*(C_UDSI(c,t,UU)*C_DUDX(c,t)+C_UDSI(c,t,UV)*C_DVDX(c,t)+
C_UDSI(c,t,UW)*C_DWDX(c,t)+C_UDSI(c,t,UV)*C_DUDY(c,t)+C_UDSI(c,t,VV)*C_DVDY(c,t)+C_UDSI(c,t,VW)*C_DWD
Y(c,t)+C_UDSI(c,t,UW)*C_DUDZ(c,t)+C_UDSI(c,t,VW)*C_DVDZ(c,t) + C_UDSI(c,t,WW)*C_DWDZ(c,t));
return prod;
}
DEFINE_SOURCE(e_source, c, t, dS, eqn)
{
real prod;
real c_1e =1.44;
dS[eqn]= 0.0;
prod=                    -                    c_1e*C_D(c,t)/C_K(c,t)*C_R(c,t)*(C_UDSI(c,t,UU)*C_DUDX(c,t)+
C_UDSI(c,t,UV)*C_DVDX(c,t)+C_UDSI(c,t,UW)*C_DWDX(c,t)+C_UDSI(c,t,UV)*C_DUDY(c,t)+C_UDSI(c,t,VV)*C_DVD
Y(c,t)+C_UDSI(c,t,VW)*C_DWDY(c,t)+C_UDSI(c,t,UW)*C_DUDZ(c,t)    +    C_UDSI(c,t,VW)*C_DVDZ(c,t)    +
```

```
C_UDSI(c,t,WW)*C_DWDZ(c,t));
return prod;
}
// Define Eddy  viscosity
DEFINE_TURBULENT_VISCOSITY(Craft_Mu_t,c,t)
{
real S,W,c_mu,T,mu_t;
real S11, S12, S13, S21, S22, S23, S31, S32, S33;
real W11, W12, W13, W21, W22, W23, W31, W32, W33;
// Define variable
real u = C_MU_L(c,t); //laminar viscosity
real p   = C_R(c,t);      //density
real v, Ret,fmu;
real k   = C_K(c,t);      //Turbulent kinetic energy(k)
real e   = C_D(c,t);      //Dissipation rate (epsilon)
              T=C_K(c,t)/C_D(c,t); // k/e
// Define stress tensor
                                                S11 = 0.5*( C_DUDX(c,t)+C_DUDX(c,t) );
                                                S12 = 0.5*( C_DUDY(c,t)+C_DVDX(c,t) );
                                                S13 = 0.5*( C_DUDZ(c,t)+C_DWDX(c,t) );
                                                S21 = 0.5*( C_DVDX(c,t)+C_DUDY(c,t) );
                                                S22 = 0.5*( C_DVDY(c,t)+C_DVDY(c,t) );
                                                S23 = 0.5*( C_DVDZ(c,t)+C_DWDY(c,t) );
                                                S31 = 0.5*( C_DWDX(c,t)+C_DUDZ(c,t) );
                                                S32 = 0.5*( C_DWDY(c,t)+C_DVDZ(c,t) );
                                                S33 = 0.5*( C_DWDZ(c,t)+C_DWDZ(c,t) );
S=T*sqrt( 2*(S11*S11+S12*S12+S13*S13+S21*S21+S22*S22+S23*S23+S31*S31+S32*S32+S33*S33) );
// Define vorticity tensor
                                                W11 = 0.5*( C_DUDX(c,t)-C_DUDX(c,t) );
                                                W12 = 0.5*( C_DUDY(c,t)-C_DVDX(c,t) );
                                                W13 = 0.5*( C_DUDZ(c,t)-C_DWDX(c,t) );
                                                W21 = 0.5*( C_DVDX(c,t)-C_DUDY(c,t) );
                                                W22 = 0.5*( C_DVDY(c,t)-C_DVDY(c,t) );
                                                W23 = 0.5*( C_DVDZ(c,t)-C_DWDY(c,t) );
                                                W31 = 0.5*( C_DWDX(c,t)-C_DUDZ(c,t) );
                                                W32 = 0.5*( C_DWDY(c,t)-C_DVDZ(c,t) );
```

```
W33 = 0.5*( C_DWDZ(c,t)-C_DWDZ(c,t) );
W=T*sqrt( 2*(W11*W11+W12*W12+W13*W13+W21*W21+W22*W22+W23*W23+W31*W31+W32*W32+W33*W33) );
// Define kinenatic viscosity
v = u/p;
// Define turbulent Reynolds number
Ret=k*k/(e*v);
// Define Constant Cmu
c_mu=(0.3/(1.0+(0.35* MAX(S,W) )))*(1.0-exp(-0.36/(exp(-0.75* MAX(S,W) ))));
// Define Damping function equal 1.0 for High Reynolds number
fmu=1.;
// Define Eddy viscosity
mu_t=c_mu*T*C_K(c,t);
return mu_t;
}
DEFINE_ADJUST(rsm_adjust,domain)
{
Thread  *t;
cell_t  c;
real X;
real mu_t, c_mu;
real S11, S12, S13, S21, S22, S23, S31, S32, S33;
real W11, W12, W13, W21, W22, W23, W31, W32, W33;
real S, W, Sij, Wij;
real C_4, C_6, C_7;
real P11, P22, P33, Pkk;
real tau_w, u_tauw, u_star, y_star, Gk, u_mag;
real u ;
real p ;
real v, Ret,fmu;
real k ;
real e ;
/* Set the turbulent viscosity */
thread_loop_c(t,domain)
if (FLUID_THREAD_P(t))
{
begin_c_loop(c,t)
```

```
{
// Define variable
u= C_MU_L(c,t); //laminar viscosity
p= C_R(c,t);            //density
k= C_K(c,t);            //Turbulent kinetic energy(k)
e= C_D(c,t);            //Dissipation rate (epsilon)
X=C_K(c,t)/C_D(c,t); // k/e
// Define stress tensor
                                        S11 = 0.5*( C_DUDX(c,t)+C_DUDX(c,t) );
                                        S12 = 0.5*( C_DUDY(c,t)+C_DVDX(c,t) );
                                        S13 = 0.5*( C_DUDZ(c,t)+C_DWDX(c,t) );
                                        S21 = 0.5*( C_DVDX(c,t)+C_DUDY(c,t) );
                                        S22 = 0.5*( C_DVDY(c,t)+C_DVDY(c,t) );
                                        S23 = 0.5*( C_DVDZ(c,t)+C_DWDY(c,t) );
                                        S31 = 0.5*( C_DWDX(c,t)+C_DUDZ(c,t) );
                                        S32 = 0.5*( C_DWDY(c,t)+C_DVDZ(c,t) );
                                        S33 = 0.5*( C_DWDZ(c,t)+C_DWDZ(c,t) );
S=X*sqrt( 2*(S11*S11+S12*S12+S13*S13+S21*S21+S22*S22+S23*S23+S31*S31+S32*S32+S33*S33) );
// Define vorticity tensor
                                        W11 = 0.5*( C_DUDX(c,t)-C_DUDX(c,t) );
                                        W12 = 0.5*( C_DUDY(c,t)-C_DVDX(c,t) );
                                        W13 = 0.5*( C_DUDZ(c,t)-C_DWDX(c,t) );
                                        W21 = 0.5*( C_DVDX(c,t)-C_DUDY(c,t) );
                                        W22 = 0.5*( C_DVDY(c,t)-C_DVDY(c,t) );
                                        W23 = 0.5*( C_DVDZ(c,t)-C_DWDY(c,t) );
                                        W31 = 0.5*( C_DWDX(c,t)-C_DUDZ(c,t) );
                                        W32 = 0.5*( C_DWDY(c,t)-C_DVDZ(c,t) );
                                        W33 = 0.5*( C_DWDZ(c,t)-C_DWDZ(c,t) );
W=X*sqrt( 2*(W11*W11+W12*W12+W13*W13+W21*W21+W22*W22+W23*W23+W31*W31+W32*W32+W33*W33) );
// Define kinenatic viscosity
v = u/p;
// Define turbulent Reynolds number
Ret=k*k/(e*v);
// Define Constant Cmu
c_mu=(0.3/(1.0+(0.35* MAX(S,W) )))*(1.0-exp(-0.36/(exp(-0.75* MAX(S,W) ))));
// Define Damping function equal 1.0 for High Reynolds number
```

```
fmu=1.;
// Define Eddy viscosity
mu_t=c_mu*fmu*C_K(c,t)*C_K(c,t)*C_D(c,t);
// Define constant of Craft et al.(1996)
C_4=-10.0*SQR(c_mu);
C_6=-5.0*SQR(c_mu);
C_7=5.0*SQR(c_mu);
// Define tensor index = SklSkl
Sij=S11*S11+S12*S12+S13*S13+S21*S21+S22*S22+S23*S23+S31*S31+S32*S32+S33*S33;
// Define tensor index = WklWkl
Wij=W11*W11+W12*W12+W13*W13+W21*W21+W22*W22+W23*W23+W31*W31+W32*W32+W33*W33;
//*********************************************************************//
//************** IMPORTANT define only anisotropy tensor ******************//
//************** and multiply k in the Reynolds stress  ********************//
//**************  anisotropy tensor*****************************************//
//*********************************************************************//
//*********************************************************************//
// Define anisotropy tensor of normal Reynolds stress
C_UDSI(c,t,UU)=
                                        C_1*mu_t*X*  (S11*S11+S12*S12+S13*S13-1./3.*Sij)
                                       + 2.*C_2*mu_t*X*  (W12*S12+W13*S13)
                                       + C_3*mu_t*X*  (W12*W12+W13*W13-1./3.*Wij)
                                       -2.*C_4*mu_t*X*X*
(W12*(S11*S12+S12*S22+S13*S23)+W13*(S11*S13+S12*S23+S13*S33))
                                       +   C_6*mu_t*X*X* (S11*Sij)
                                       +   C_7*mu_t*X*X* (S11*Wij);

C_UDSI(c,t,VV)=
                                            C_1*mu_t*X*  (S12*S12+S22*S22+S23*S23-1./3.*Sij)
                                       + 2.*C_2*mu_t*X*  (W23*S23-W12*S12)
                                        +   C_3*mu_t*X*  (W12*W12+W23*W23-1./3.*Wij)
                                       +          2.*C_4*mu_t*X*X*          (W12*(S11*S12+S12*S22+S13*S23)-
W23*(S12*S13+S22*S23+S23*S33))
                                        +   C_6*mu_t*X*X* (S22*Sij)
                                        +   C_7*mu_t*X*X* (S22*Wij);
```

```
C_UDSI(c,t,WW)=
                                        C_1*mu_t*X*   (S13*S13+S23*S23+S33*S33-1./3.*Sij)
                                      - 2.*C_2*mu_t*X*   (W13*S13+W23*S23)
                                      + C_3*mu_t*X*   (W13*W13+W23*W23-1./3.*Wij)
                                      +2.*C_4*mu_t*X*X*
(W13*(S11*S13+S12*S23+S13*S33)+W23*(S12*S13+S22*S23+S23*S33))
                                      +    C_6*mu_t*X*X* (S33*Sij)
                                      +    C_7*mu_t*X*X* (S33*Wij);
// Define anisotropy tensor of shear Reynolds stress
        C_UDSI(c,t,UV)=
                                        C_1*mu_t*X*   (S11*S12+S12*S22+S13*S23)
                                      +    C_2*mu_t*X*   (W12*(S22-S11)+W13*S23+W23*S13)
                                      +    C_3*mu_t*X*   (W13*W23)
                                      +          C_4*mu_t*X*X*          (W12*(S11*S11+S13*S13-S22*S22-S23*S23)-
W13*(S12*S13+S22*S23+S23*S33)-W23*(S11*S13+S12*S23+S13*S33))
                                      +    C_6*mu_t*X*X* (S12*Sij)
                                      +    C_7*mu_t*X*X* (S12*Wij);

        C_UDSI(c,t,UW)=
                                        C_1*mu_t*X*   (S11*S13+S12*S23+S13*S33)
                                      +    C_2*mu_t*X*   (W13*(S33-S11)+W12*S23-W23*S12)
                                      -    C_3*mu_t*X*   (W12*W23)
                                      +                C_4*mu_t*X*X*     (W13*(S11*S11+S12*S12-S23*S23-S33*S33)-
W12*(S11*S13+S22*S23+S23*S33)+W23*(S11*S12+S12*S22+S13*S23))
                                      +    C_6*mu_t*X*X* (S13*Sij)
                                      +    C_7*mu_t*X*X* (S13*Wij);

        C_UDSI(c,t,VW)=
                                        C_1*mu_t*X*   (S12*S13+S22*S23+S23*S33)
                                      +    C_2*mu_t*X*   (W23*(S33-S22)-W12*S13-W13*S12)
                                      +    C_3*mu_t*X*   (W12*W13)
                                      +                        C_4*mu_t*X*X*       (W23*(S12*S12+S22*S22-S13*S13-
S33*S33)+W12*(S11*S13+S12*S23+S13*S33)+W13*(S11*S12+S12*S22+S13*S23))
                                      +    C_6*mu_t*X*X* (S23*Sij)
                                      +    C_7*mu_t*X*X* (S23*Wij);
// Define memory Reynolds stress by User defined Memory
```

```
//*************************************************************************//
//**************IMPORTANT define Linear and Nonlinear Term*****************//
//*************************************************************************//
C_UDMI(c,t,0)=        2./3.*C_K(c,t) -2.*mu_t*S11  + C_UDSI(c,t,UU);
C_UDMI(c,t,1)=        2./3.*C_K(c,t) -2.*mu_t*S22  + C_UDSI(c,t,VV);
C_UDMI(c,t,2)=        2./3.*C_K(c,t) -2.*mu_t*S33  + C_UDSI(c,t,WW);
C_UDMI(c,t,3)=       -2.*mu_t*S12  + C_UDSI(c,t,UV);
C_UDMI(c,t,4)=       -2.*mu_t*S13  + C_UDSI(c,t,UW);
C_UDMI(c,t,5)=       -2.*mu_t*S23  + C_UDSI(c,t,VW);
}
end_c_loop(c,t)
}
}
```