

Introduzione alla fluidodinamica computazionale

La fluidodinamica computazionale consente di modellare a livello teorico il campo di velocità di un fluido che attraversa un sistema di qualunque forma del quale si conoscano le condizioni al contorno. Può essere inoltre utilizzata per la valutazione dello scambio di calore effettuato dal fluido e più in generale per valutazioni energetiche.

Il software Fluent è un codice CFD, acronimo di *computational fluid dynamics*, del tipo *general purpose*, cioè in grado di descrivere una vasta gamma di casi mediante tecniche di discretizzazione delle equazioni di Navier-Stokes.

La caratteristica saliente del programma è la possibilità di descrivere un qualunque campo di moto sia esso bidimensionale o tridimensionale fornendo dei risultati che di rado sono messi a disposizione dai modelli analitici noti.

Infatti, normalmente, gli studi fluidodinamici bidimensionali o molto più spesso tridimensionali richiedono la costruzione di modelli del sistema da studiare in scala e misurazioni sperimentali utili a formulare una descrizione adimensionalizzata dei fenomeni. Il difetto di tali modelli è la loro relativa “ermeticità”, nel senso che consentono agevoli valutazioni quantitative ma non fanno comprendere ciò che avviene all’interno del sistema.

Un’alternativa all’analisi dimensionale è la misurazione mediante opportuni di strumenti di misura delle velocità istantanee raggiunte dal fluido quando attraversa il modello. Le misure sperimentali forniscono indicazioni teoriche molto più significative ma richiedono anche molto tempo e cospicui investimenti in strumentazioni di misura, a fronte di un campo di validità ristretto alle singole sperimentazioni.

I codici di calcolo CFD permettono invece la simulazione del campo di moto all’interno di un’oggetto di qualunque forma, con un tempo di calcolo dipendente solo dalle capacità dell’elaboratore e quindi mai eccessivamente lungo, se si sfruttano le notevoli capacità di calcolo delle postazioni più recenti.

Si consideri come esempio la simulazione dei gas allo scarico di un missile che rappresenta lo studio di un fenomeno turbolento, quindi descrivibile solo tramite grandezze medie e non certo istantanee, a sua volta non stazionario. Oltretutto questo tipo di simulazione fluidodinamica è strettamente intrecciata con lo scambio termico così come dimostrato dall’analogia di Reynolds. Data la dimensione computazionale del problema già quando è posto in forma bidimensionale, legata essenzialmente alle dimensioni esterne della geometria da analizzare, e la grande difficoltà a risolvere per via analitica le equazioni differenziali di Navier-Stokes, l’unica soluzione al problema sarebbe stata la costruzione di un modello in scala per realizzare uno studio sperimentale. Invece i metodi computazionali per l’analisi al computer delle equazioni di Navier-Stokes in forma discretizzata, hanno consentito uno studio teorico della fluidodinamica dei gas di scarico. La fig.1 rappresenta i vettori velocità relativi ai gas allo scarico di un missile in partenza da una piattaforma di lancio, ottenuti con un modello bidimensionale [3].

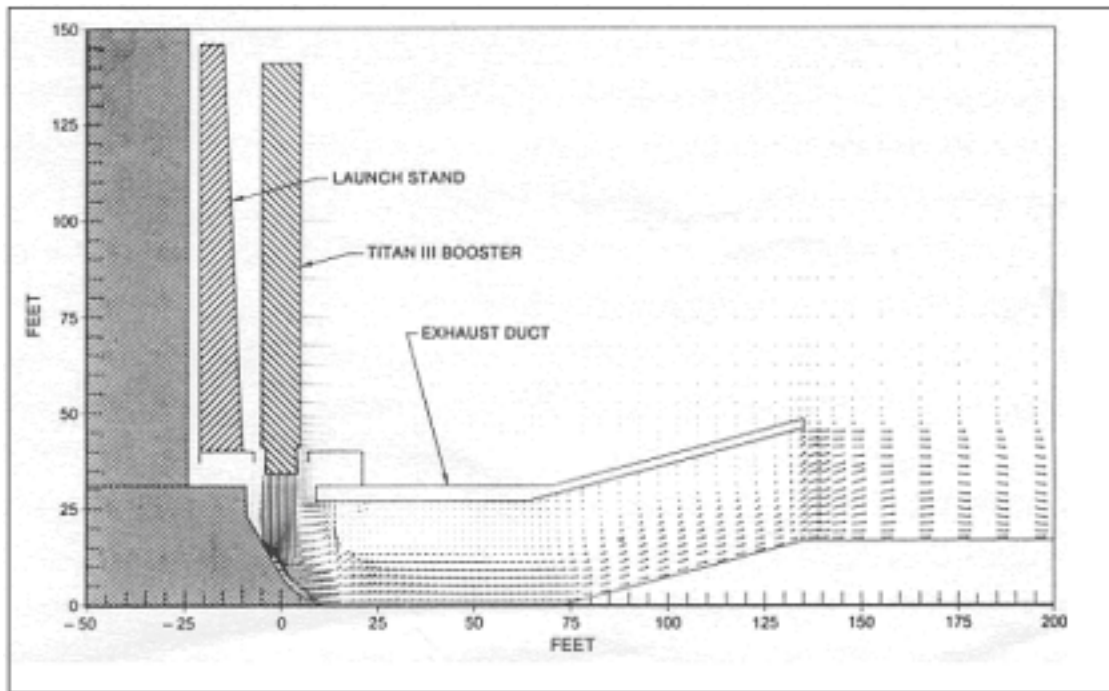


Fig. 1 Vettori velocità relativi ai gas allo scarico di un missile

Quindi, a fronte di un investimento limitato in rapporto ai vantaggi ottenuti, tali programmi offrono la possibilità di analizzare problemi fluidodinamici non risolti in passato per via dei problemi di investimento e di tempi di realizzazione annessi alle sperimentazioni.

Gli svantaggi delle simulazioni virtuali sono la presenza di errori legati alla discretizzazione dei sistemi continui, che è indispensabile per la loro analisi al calcolatore, e la necessità di far acquisire maggiori competenze agli utenti dei programmi.

In ultima analisi una volta acquisite le conoscenze necessarie per utilizzare questi programmi, in modo da ridurre in maniera ragionevole gli errori, è possibile simulare il comportamento di un oggetto a costi minimi e in tempi estremamente rapidi ed effettuare una validazione sperimentale solo sulla geometria finale. In tal modo il compito di ottimizzare una geometria per via iterativa, non viene più svolto dalle analisi sperimentali ma dai calcolatori, con un notevole risparmio in termini di *time to market* che, come noto, condiziona in maniera pesante le scelte operative relative allo sviluppo dei prodotti.

La struttura dei codici di calcolo CFD prevede di solito un programma di modellazione bidimensionale e tridimensionale, per effettuare la fase di *preprocessing*, e un solutore che poi effettua anche la fase di *postprocessing* cioè consente l'analisi dei risultati.

La fase di *preprocessing* consiste nella formulazione di un modello discreto costituito da una *mesh*, cioè una griglia di tanti elementi che approssima la geometria reale. Ovviamente non sempre il modello costruito è la copia del sistema da studiare, in quanto la presenza di simmetrie permette di semplificarlo ed inoltre, in ambito fluidodinamico, è sempre opportuno definire i contorni del sistema da studiare laddove sia possibile stabilire condizioni al contorno che abbiano senso.

Nell'ambito degli studi di questa tesi, il programma usato per effettuare la fase di *preprocessing* è Gambit, ed è strettamente integrato con Fluent, che è il solutore, nel senso che Fluent potrebbe importare le mesh anche da pacchetti CAD esterni, ma quest'operazione richiederebbe operazioni intermedie di conversione, peraltro non prive di errori, mentre con Gambit è possibile esportare direttamente la mesh in Fluent.

Il metodo di analisi utilizzato da Fluent è quello dei volumi finiti, ed è importante osservare subito che questo metodo, come quello degli elementi finiti, è affetto da errori essenzialmente dipendenti dalla necessità di approssimare il comportamento di un sistema continuo con un modello discreto, con un'errore dipendente dal grado di affinamento della *mesh* nelle

zone più critiche, laddove i gradienti delle grandezze studiate sono maggiori. Una differenza importante è però costituita dalla dipendenza che i risultati hanno dalla strategia di convergenza impostata e anche dal livello di convergenza assunto come accettabile.

La ragione per la quale sussiste tale differenza è che negli elementi finiti la soluzione del problema è data dalla formulazione :

$$[A] \cdot (X) = (B) \quad (1)$$

Ove A indica la matrice di rigidezza della struttura B il vettore dei carichi nodali ed X il vettore degli spostamenti nodali tramite il quale è possibile, in seguito, ricostruire il campo delle tensioni nella struttura. Con il metodo dei volumi finiti si arriva mediante passaggi matematici ad una formulazione identica a quella degli elementi finiti, ma nella quale matrici e vettori non hanno nessun significato fisico diretto, eccezion fatta per il vettore incognita che è il vettore delle velocità.

Anche le equazioni di Navier-Stokes, con le grandezze fisiche espresse come funzione continua dello spazio e del tempo, vengono discretizzate mediante il metodo delle differenze finite spaziali e temporali, e tramite ulteriori passaggi matematici si può ricondurre il problema alla forma espressa nella (1). Dato che poi la matrice A viene ad assumere grandi dimensioni, si pone il problema di memorizzare la matrice in un calcolatore, per invertirla minimizzando la quantità di memoria usata. Per questo, tramite opportune trasformazioni, tutti gli zeri della matrice vengono concentrati in blocchi contigui, in modo da evitarne la memorizzazione.

Questo non è però ancora sufficiente a risolvere il problema dell'eccessiva quantità di informazioni da memorizzare e allora si sceglie di risolvere il problema per via iterativa così come mostrato dalla (2).

$$[P] \cdot ((X^{(k+1)}) - (X^{(k)})) = \alpha((B) - [A] \cdot (X^{(k)})) \quad (2)$$

Nella notazione introdotta P è una matrice meglio nota come preconditionatore, $X^{(k)}$ è il vettore delle incognite all'iterazione k e α rappresenta il coefficiente di rilassamento.

La formula (2) mostra come la soluzione del sistema avviene in modo ben diverso da come viene fatto nel metodo degli elementi finiti. Infatti il sistema di equazioni viene risolto cercando di approssimare il vettore soluzione del problema per via iterativa e non invertendo la matrice A , che rappresenta l'equivalente della matrice di rigidezza della struttura degli elementi finiti.

Appare quindi evidente che la soluzione dipende strettamente dal numero delle iterazioni che vengono effettuate e dal criterio che viene stabilito per assumere la soluzione convergente, perché il vettore delle incognite all'iterazione k , è solo un vettore di prova con il quale si tenta di individuare una possibile soluzione, minimizzando gli scarti rispetto alla soluzione esatta. La definizione di un criterio di convergenza, inteso come il rapporto tra il valore assunto da una variabile nell'iterazione corrente e il valore che essa aveva nelle prime iterazioni, è del tutto arbitraria.

Un ruolo non marginale viene svolto dal coefficiente di rilassamento α la cui definizione è indispensabile per la convergenza della soluzione e quindi per la soluzione di un generico caso da studiare e il suo valore è sempre inferiore all'unità .

Nel capitolo 1 viene descritta la GUI (*Graphic user interface*) dei due programmi Gambit e Fluent e nei capitoli successivi vengono presentati alcuni casi semplici dei quali se ne conosce la soluzione analitica.

L'obiettivo è ovviamente quello di verificare l'accuratezza dell'analisi numerica fornita dal software in funzione dei parametri richiesti per la simulazione. Tale esperienza può essere poi utilizzata come base per l'analisi di casi più complessi, e in questi casi sarebbe comunque necessaria una minima validazione sperimentale, a conferma delle indicazioni fornite dalla simulazione numerica.

Una precisazione doverosa va fatta sulla scelta della formulazione matematica e dei simboli utilizzati per le varie grandezze fisiche in questi capitoli ; infatti l'utilizzo del programma e la sua comprensione rientrano nell'ambito teorico degli studi sulla fluidodinamica computazionale, cioè dei metodi di analisi numerica per la soluzione al calcolatore di casi non risolvibili mediante modelli analitici.

Tali studi sono stati sviluppati recentemente negli Stati Uniti e i relativi risultati sono pervenuti in una seconda fase in Europa. Pertanto si può dire che in ambito teorico la CFD è un campo ancora aperto ed ha tuttora ulteriori margini di sviluppo, per cui è difficile reperire materiale in letteratura e i pochi riferimenti bibliografici disponibili sono tutti di origine statunitense.

CAPITOLO PRIMO

Moto laminare in un tubo : modello piano assialsimmetrico**1.1 Modello analitico del moto laminare in un tubo**

Come primo caso di studio è stato preso in considerazione il moto dell'aria in un capillare di diametro 0,5 mm e lunghezza 5 cm con un numero di Reynolds pari a 1500, e quindi il moto è laminare perché Re è ampiamente inferiore al valore 2000~2500, che è sperimentalmente indicato come soglia di transizione dal moto laminare a quello turbolento.

La teoria [1] in questo caso fornisce una chiara soluzione che deriva dall'integrazione dell'equazione di conservazione della quantità di moto applicata al caso dei moti unidirezionali, incomprimibili e stazionari (1.1).

$$\vec{\text{grad}} \cdot p = \mu \cdot \Delta_2 u \quad (1.1)$$

Nella notazione della formula precedente, p è la pressione relativa, μ è la viscosità cinematica del fluido e u è la velocità diretta lungo la direzione del moto. Inoltre Δ_2 è l'operatore di *Laplace* definito dalla (1.2).

$$\Delta_2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2} \quad (1.2)$$

In questo caso la velocità si può assumere come funzione della sola coordinata radiale r e l'operatore definito dalla (1.2) assume la forma (1.3).

$$\Delta_2 = \frac{1}{r} \cdot \frac{\partial}{\partial r} \left(r \frac{\partial u}{\partial r} \right) \quad (1.3)$$

Chiamando i la pendenza motrice definita come:

$$i = -\frac{\partial p}{\partial s} \quad (1.4)$$

ove s è la generica ascissa curvilinea lungo la traiettoria del moto, l'equazione (1.1) può essere integrata lungo la direzione radiale.

$$r \frac{\partial u}{\partial r} = -\frac{i}{\mu} \frac{r^2}{2} + c_1 \quad (1.5)$$

Nella (1.5) c_1 costituisce una costante di integrazione. Integrando ulteriormente la (1.5) rispetto alla coordinata radiale si ottiene la (1.6).

$$u(r) = -\frac{i}{\mu} \frac{r^2}{4} + c_1 \ln r + c_2 \quad (1.6)$$

Dato che per $r=0$ il termine logaritmico tende a $-\infty$ si assume la costante c_1 nulla mentre la costante c_2 si trova imponendo velocità nulla sulla parete del tubo, ove $r=D/2$, se si indica con D il diametro della tubazione. Si ottiene così la (1.7).

$$u(r) = \frac{i}{4\mu} \left(\frac{D^2}{4} - r^2 \right) \quad (1.7)$$

Tale formula descrive una distribuzione di velocità parabolica con valore massimo sull'asse della tubazione e decrescente dal centro verso le pareti ; data la simmetria radiale di tale distribuzione il profilo di velocità è in pratica un paraboloide con asse coincidente con quello del condotto.

Il valor medio di tale distribuzione di velocità è poi la metà del valore massimo come si può notare dalla seguente dimostrazione:

$$u(r) = \frac{i}{4\mu} \left(\frac{D^2}{4} - r^2 \right) = c_3 \cdot (c_4 - r^2) \quad (1.8)$$

avendo assunto:

$$c_3 = \frac{i}{4\mu}; c_4 = \frac{D^2}{4} \quad (1.9)$$

Posto $u = cost. = c_5$:

$$c_5 = c_3 \cdot (c_4 - r^2) \Rightarrow r = \sqrt{c_4 - \frac{c_5}{c_3}} = \sqrt{\frac{D^2}{4} - \frac{u}{c_3}} \quad (1.10)$$

Il volume di un paraboloide dato dalla precedente distribuzione di velocità si può determinare analiticamente tramite un integrale di volume calcolato per strati.

$$V = \int_0^{u_{max}} \left[\pi \left(\frac{D^2}{4} - \frac{u}{c_3} \right) \right] du = \frac{\pi \cdot u_{max}}{2} \left(\frac{D^2}{2} - \frac{u_{max}}{c_3} \right) \quad (1.11)$$

$$V = \frac{\pi \cdot D^2}{4} \cdot u_m \Rightarrow u_m = u_{max} \cdot \left(1 - \frac{u_{max} \cdot 2}{c_3 \cdot D^2} \right) \quad (1.12)$$

Definendo allora la quantità:

$$\frac{u_m}{u_{max}} = \frac{u_{max}}{c_3} = \frac{u_{max}}{\frac{16 \cdot \mu}{4 \cdot \mu}} = \frac{16 \cdot \mu}{4 \cdot \mu} = \frac{D^2}{4} \quad (1.13)$$

Si ottiene :

$$u_m = u_{max} \left(1 - \frac{D^2}{4} \cdot \frac{2}{D^2} \right) = \frac{1}{2} u_{max} \quad (1.14)$$

Il valor medio della distribuzione è allora la metà del valor massimo come si può osservare dal grafico di fig. 1.1.

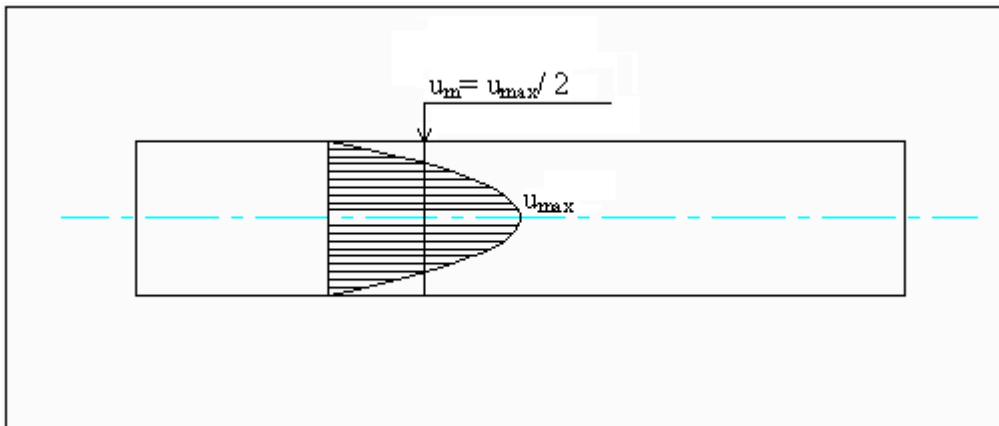


Fig. 1.1 Profilo di velocità teorico in un moto laminare

1.2 Cenni preliminari all'utilizzo di Gambit e Fluent

Per effettuare una verifica del modello con Fluent occorre prima di tutto realizzare un modello del fenomeno con Gambit, cioè una *mesh* che approssima la geometria presa in esame. Per poter comprendere come utilizzare Gambit e il solutore Fluent è necessario però comprendere prima i tipi di file generati.

In ambiente Unix bisogna lanciare i programmi mediante il *prompt* dei comandi. Per lanciare Gambit occorre digitare la seguente riga di comando:

```
Prompt>Gambit -id file -new
```

Ove “-id” indica un generico identificatore della sessione corrente (“-“ precede tutte le opzioni nella sintassi della riga di comando), *file* indica il nome del file che si sta aprendo non preceduto da alcuna estensione e “-new” indica che il file che si sta aprendo è nuovo ; in caso il file sia già esistente occorre utilizzare l’opzione “-old”.

Da quanto detto appare chiaro che non si può far partire Gambit se prima non si conosce il nome del *file* da aprire sia esso già esistente o nuovo. Bisogna però far molta attenzione nel distinguere i vari *file* che Gambit gestisce. Infatti se il *file* si chiama ad esempio tubo allora nella directory di lavoro dell’utente saranno presenti i seguenti file al termine della sessione con Gambit, dopo aver salvato il lavoro :

```
tubo.jou
tubo.trn
tubo.dbs
tubo.lok
```

I primi tre *file* contengono le informazioni relative al lavoro svolto e sono quindi indispensabili per ricostruire il modello in una successiva sessione di Gambit mentre il quarto contiene informazioni relative all’utilizzo di Gambit durante la sessione nella quale è stato creato il *file*.

Gambit lavora in maniera stabile ma ha il difetto di essere molto pesante e di non consentire sempre l'agevole interruzione del processo caricato. Allora in questo caso bisogna intervenire aprendo una nuova finestra relativa al *prompt* dei comandi e digitare:

```
prompt> ps -ef
```

A questo punto comparirà una lista di tutti i processi in corso e tra questi bisogna “uccidere” quello creato da Gambit situato nella *directory utenti/app/Fluent_inc* il cui nome può ovviamente cambiare a seconda della fantasia del sistemista che ha installato il software. Il comando che consente di terminare un'applicazione è :

```
prompt> kill PID
```

ove PID indica l'identificatore del processo aperto.

Se si “uccide” una sessione aperta di Gambit, che per ipotesi può essere quella relativa ai *file* tubo, allora bisogna anche inspiegabilmente eliminare il *file* tubo.lok altrimenti Gambit al successivo tentativo di aprire il *file* dirà che il *file* è stato già aperto e ve ne impedirà di fatto l'accesso.

Per eliminare il *file* digitare :

```
prompt> rm tubo.lok
```

Se invece si vuole cancellare il modello tubo allora bisogna cancellare manualmente i tre file con l'estensione *.jou, *.trn e *.dbs.

Occorre fare inoltre attenzione al fatto che per far leggere un *file* a Fluent bisogna esportare la *mesh* creata da Gambit in formato *.msh e quindi nella directory di lavoro verrà salvato anche il file tubo.msh ; pertanto cancellando i *file* tubo.dbs, tubo.jou e tubo.trn non si potrà più modificare il modello tubo sotto Gambit ma si potrà leggere la *mesh* tubo.msh per risolverla ad esempio utilizzando altre opzioni.

Fluent a sua volta consente di salvare i risultati utilizzando due ulteriori formati che sono *.cas e *.dat. In altre parole al termine di una sessione di lavoro con Fluent, nella *directory* di lavoro saranno presenti due ulteriori *file* : tubo.cas e tubo.dat. Tali file possono essere agevolmente riaperti da Fluent quando si vogliono rivedere i risultati di una certa simulazione o continuare la simulazione tramite ulteriori iterazioni, ma possono essere utilizzati anche per effettuare una nuova inizializzazione del campo di moto e quindi una nuova simulazione anche con differenti condizioni al contorno. L'utilità del *file* in formato *.msh è quindi solo quella di consentire una prima lettura della *mesh* da parte di Fluent. Leggendo i *file* *.dat e *.cas si può poi anche visualizzare graficamente la *mesh* importata.

Per una rapida e sintetica interpretazione del contenuto del paragrafo si consideri la tabella 1.1.

Tabella 1.1 Caratteristiche dei *file* generati da Gambit e Fluent

Formato	Caratteristiche
.dbs,.trn, *.jou	<i>File</i> creati da Gambit indispensabili per riaprire la <i>mesh</i> relativa ad un certo caso e modificarla
*.lok	<i>File</i> che è possibile eliminare senza problemi e che è necessario eliminare quando non è più possibile riaprire i 3 file relativi al caso a causa della chiusura della precedente sessione con il comando <i>kill</i>
*.msh	<i>File</i> che contiene la <i>mesh</i> creata da Gambit in modo che sia esportabile per la prima volta in Fluent
*.cas	<i>File</i> che contiene le informazioni relative alle impostazioni utilizzate da Fluent nella risoluzione di un certo caso. insieme al <i>file</i> *.dat può essere riaperto una seconda volta per continuare una simulazione o azzerarne i dati per incominciare una nuova con la medesima <i>mesh</i>
*.dat	<i>File</i> che contiene i dati relativi ad una simulazione con Fluent. Può essere utilizzato solo in abbinamento al corrispondente *.cas

1.3 Utilizzo del modellatore Gambit

In questa sessione viene spiegato il funzionamento di Gambit e Fluent schermata per schermata, in modo da definire a livello generale come creare ed analizzare un modello. La sessione di Gambit, così come quella di Fluent, è stata chiamata *visual*.

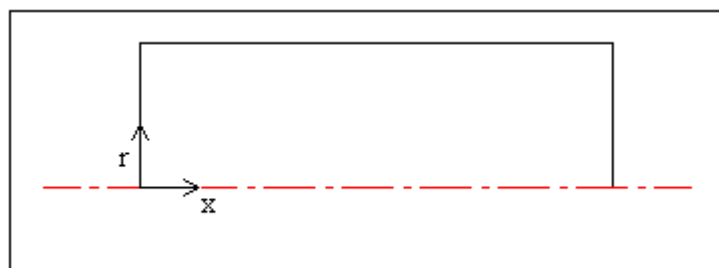
1.3.1 Descrizione generale di Gambit

Gambit è il modellatore che consente di creare dei file in formato *.msh da far leggere a Fluent ; una sessione con Gambit consta sempre di quattro fasi:

- 1) Definizione di una geometria che faccia da contorno al modello
- 2) Definizione di una *mesh* interna a questa geometria
- 3) Definizione del solutore con cui analizzare la *mesh*, delle condizioni al contorno e dell'esportazione della *mesh*

In questo caso si suppone di voler analizzare il caso del tubo percorso da fluido in moto laminare descritto nel §1.1. Data la simmetria radiale del problema è inutile studiarlo con un modello tridimensionale che darebbe gli stessi risultati di un modello piano assialsimmetrico.

Pertanto, utilizzando le indicazioni della guida di Fluent si può costruire il modello piano mostrato in fig. 1.2.

**Fig. 1.2** Disposizione del sistema di riferimento nel modello

Infatti la guida [4] suggerisce di far coincidere l'asse del modello con l'asse x del sistema di riferimento. Il rettangolo deve avere quindi la base disposta lungo questo asse e l'altezza lungo y .

La terza coordinata in direzione z viene utilizzata solo nel caso in cui si voglia creare un modello tridimensionale e quando richiesta dai menù va sempre posta a zero nei casi bidimensionali.

1.3.2 Creazione del modello piano assialsimmetrico con Gambit

Bisogna innanzitutto far partire Gambit con il comando:

```
Prompt>Gambit -id visual -new
```

Apparirà quindi la fig. 1.3.

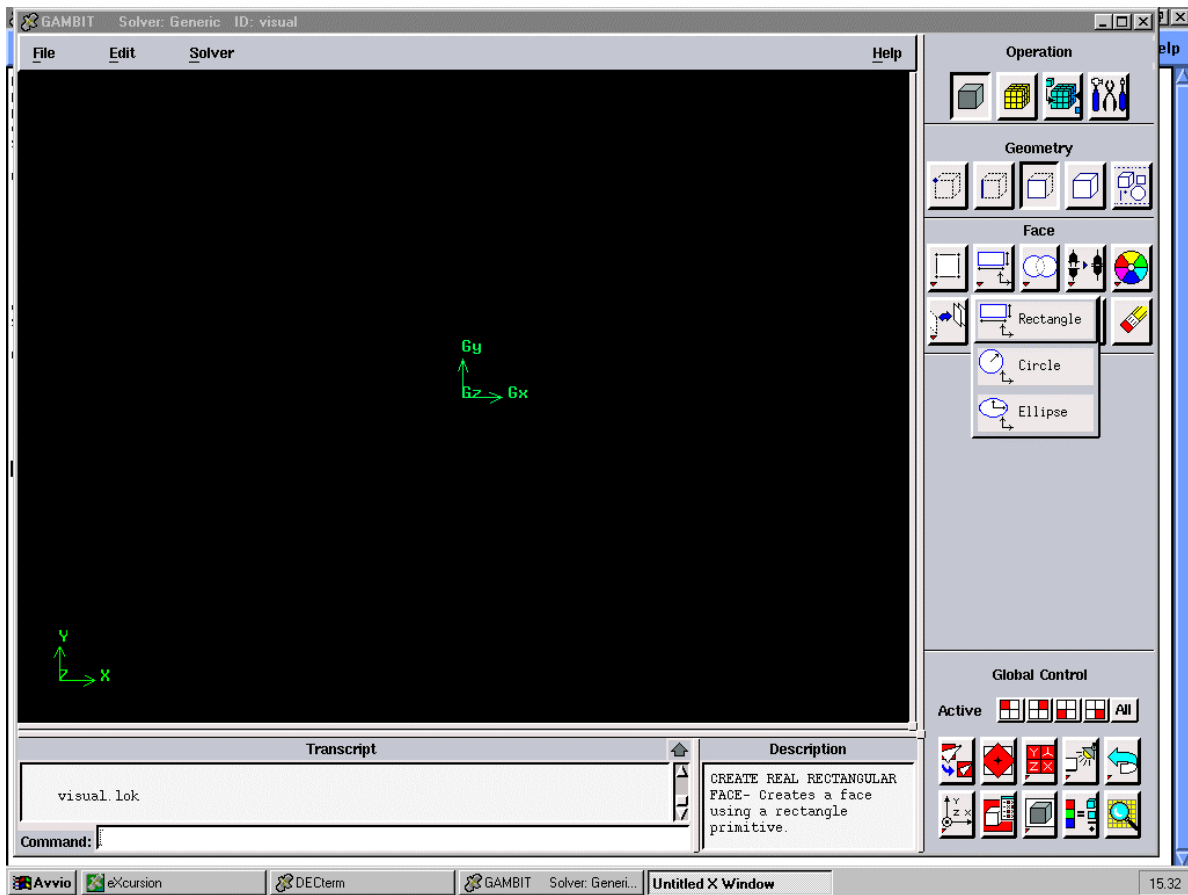


Fig. 1.3 Interfaccia utente di Gambit

Per iniziare a creare un modello piano costituito da un rettangolo bisogna prima creare i punti, poi le linee e quindi unirle per formare un'unica entità.

Dal tasto *create geometry*, in alto a destra, selezionare *create real vertex* (fig. 1.4).

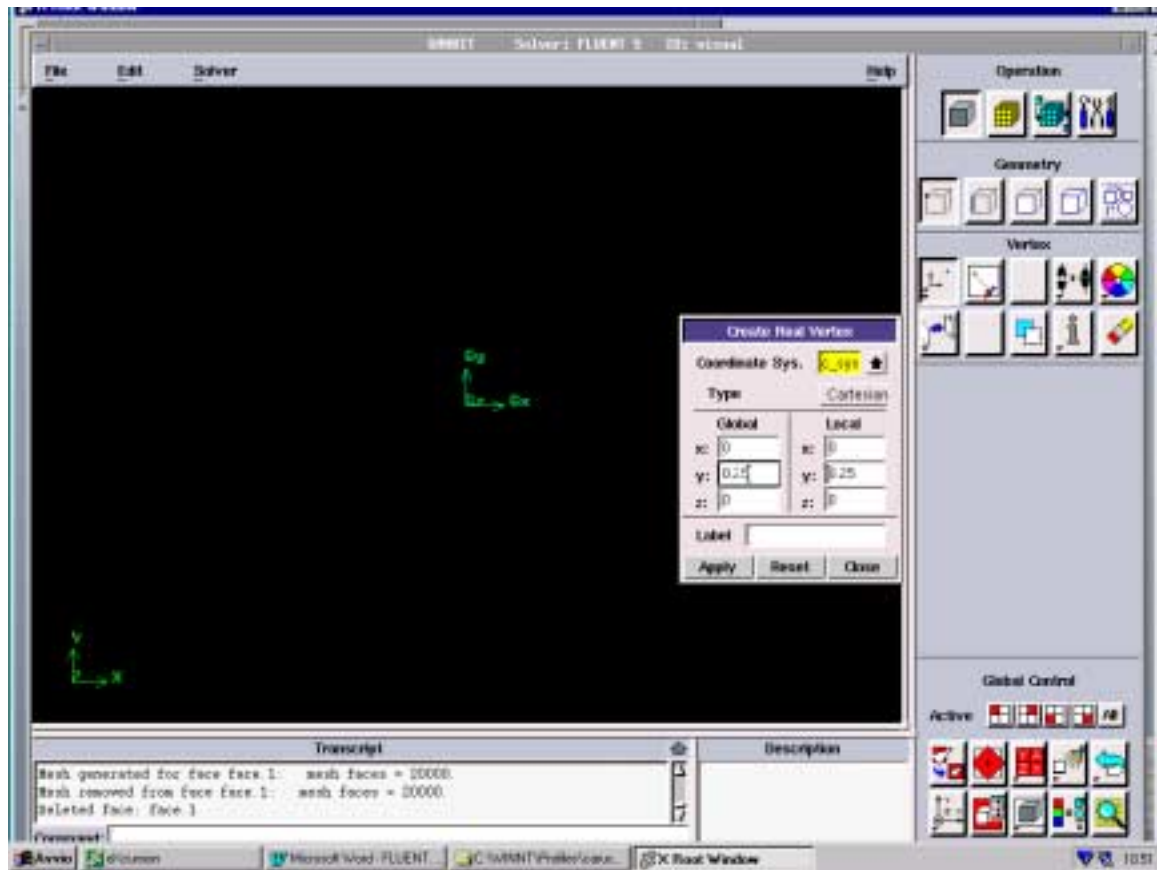


Fig. 1.4 Pulsante che in Gambit consente la creazione di un vertice

Naturalmente vengono subito chieste le coordinate del punto che si vuole creare rispetto al sistema di riferimento rappresentato in figura, e nell'esempio vengono indicate le coordinate del punto del rettangolo in alto a sinistra.

Senza chiudere la finestra è possibile creare altri punti in quanto in tutte le finestre di Gambit, una volta digitato *apply*, vengono automaticamente azzerati i dati nelle rispettive caselle per una nuova immissione. Quindi occorre digitare nelle caselle relative alle coordinate le seguenti coppie di valori:

X	Y
0	0
50	0
50	0,25

A questo punto sullo schermo saranno presenti i 4 punti costituenti i vertici del rettangolo avente base 50 e altezza 0,25. Infatti l'obiettivo di questa sessione di Gambit è la creazione di un modello piano assialsimmetrico di un tubo capillare di 0,5 mm di diametro.

Si noti che il rapporto tra la lunghezza e il diametro del condotto che si vuole creare è pari a 100. Una regola sperimentale stabilisce che, affinché un profilo di velocità laminare in un condotto di generica forma sia sviluppato, l'allungamento di quest'ultimo, inteso come rapporto L/D , deve soddisfare i requisiti della (1.15).

$$\frac{L}{D} \geq 0,029 \text{Re} \quad (1.15)$$

L'espressione (1.15) stabilisce che l'allungamento deve essere pari ad almeno il 2,9% del numero di Reynolds e D è il diametro se la sezione è circolare o in alternativa, se la sezione ha una forma diversa, una lunghezza equivalente al diametro espressa in termini di raggio idraulico.

In questo caso la (1.15) è ampiamente soddisfatta perché il rapporto di allungamento è pari a 100 mentre il 2,9% del numero di Reynolds è 43,5. Quindi in un rettangolo con tali dimensioni è lecito aspettarsi uno sviluppo completo del profilo laminare.

I vertici creati possono essere congiunti creando opportuni segmenti con il menù *create straight edge* (fig. 1.5).

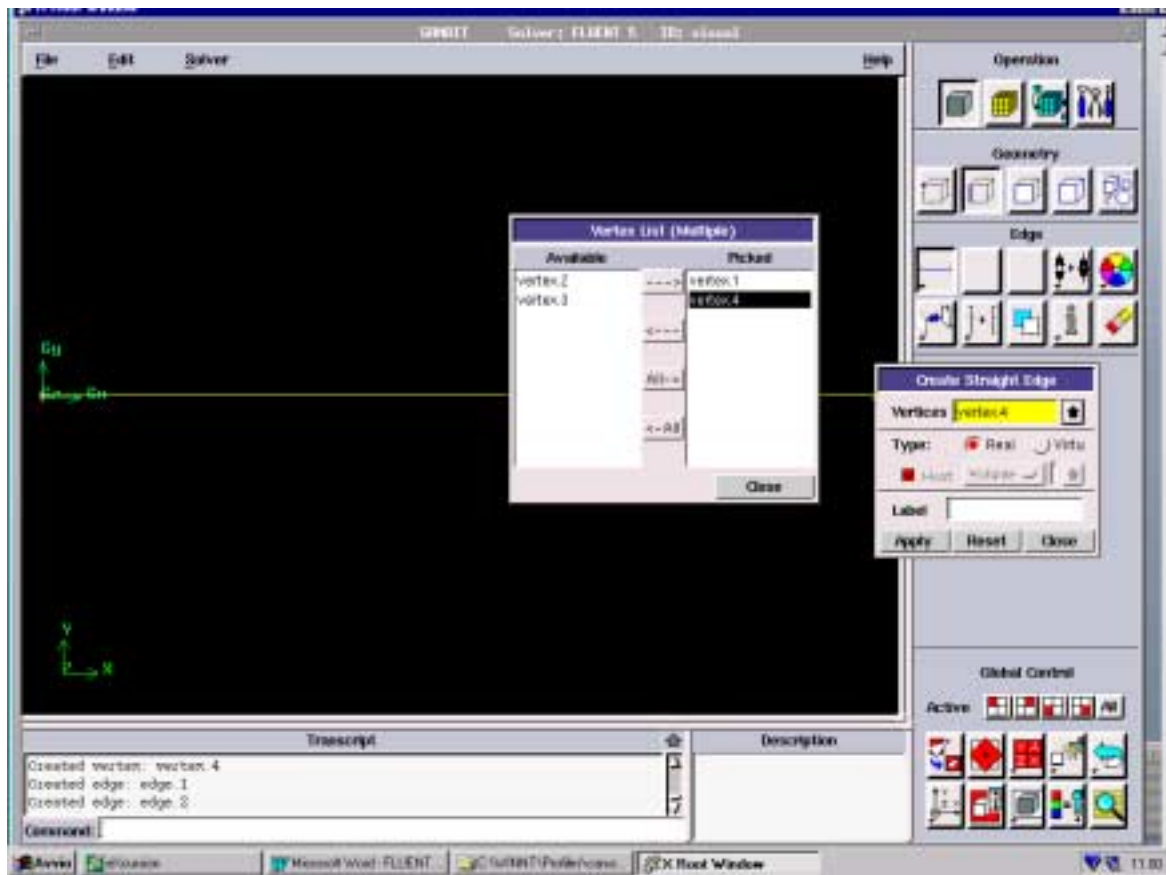


Fig. 1.5 Tasto che consente la creazione di vertici reali

Ovviamente per creare un segmento è necessario indicare i vertici che fanno da estremità selezionandoli dalla *vertex list* e cliccando su *apply*.

Per creare una faccia bisogna poi unire i vertici con il tasto *create face from wireframe* come indicato in fig. 1.6.

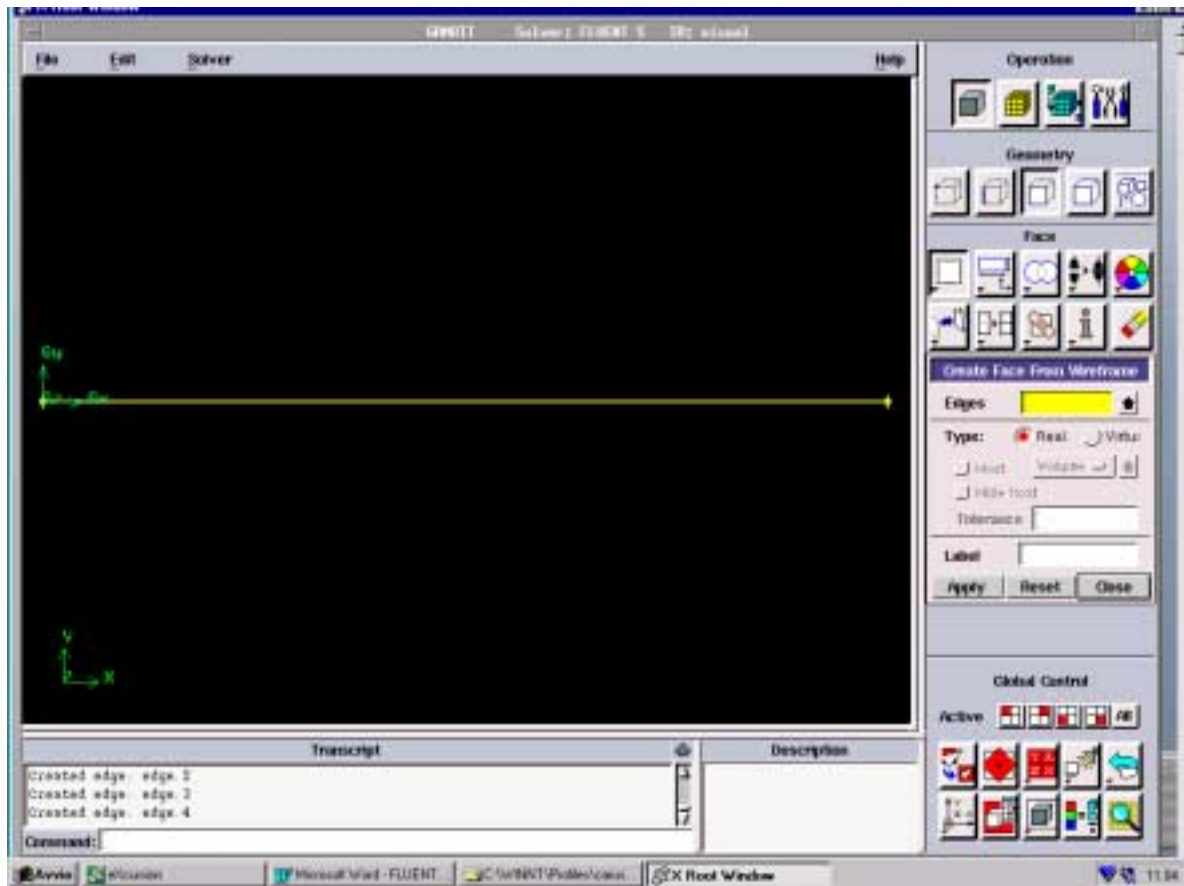


Fig. 1.6 Tasto che consente la creazione di un rettangolo da 4 segmenti separati

Bisogna quindi indicare quali segmenti occorre unire dalla lista *edges* e al solito premere *apply* per rendere effettive le modifiche e *close* per chiudere il menù.

Per creare una *mesh* su una superficie piana si possono fare due scelte alternative :

- Creare, con il pulsante *mesh faces*, una *mesh* costituita da elementi di dimensione uniforme, eventualmente distorti per essere adattati alla geometria da descrivere.
- Creare, con il pulsante *mesh edges*, una *mesh* lineare lungo i contorni della superficie in modo far generare a Gambit un numero prefissato di elementi in una certa direzione durante la creazione della *mesh* di superficie.

Dal momento che la guida di Fluent consiglia di utilizzare elementi quadrangolari allineati e allungati lungo la direzione del moto, nel caso di moti laminari sviluppati in un'unica direzione è preferibile disporre un numero di elementi non eccessivo lungo la direzione x, per poi provare al limite a vedere come varia la soluzione variando la densità degli elementi in direzione radiale.

La finestra *mesh edges* chiede di selezionare i segmenti su cui operare e il criterio per distanziare la *mesh* creata. In questo caso viene generata una *mesh* costituita da 100 elementi equispaziati utilizzando l'opzione *interval count*, e quindi nel riquadro *spacing* bisogna scrivere 100 (fig. 1.7).

Per creare la *mesh* di superficie bisogna invece usare il pulsante *mesh faces* in modo da far comparire la relativa finestra (fig. 1.8).

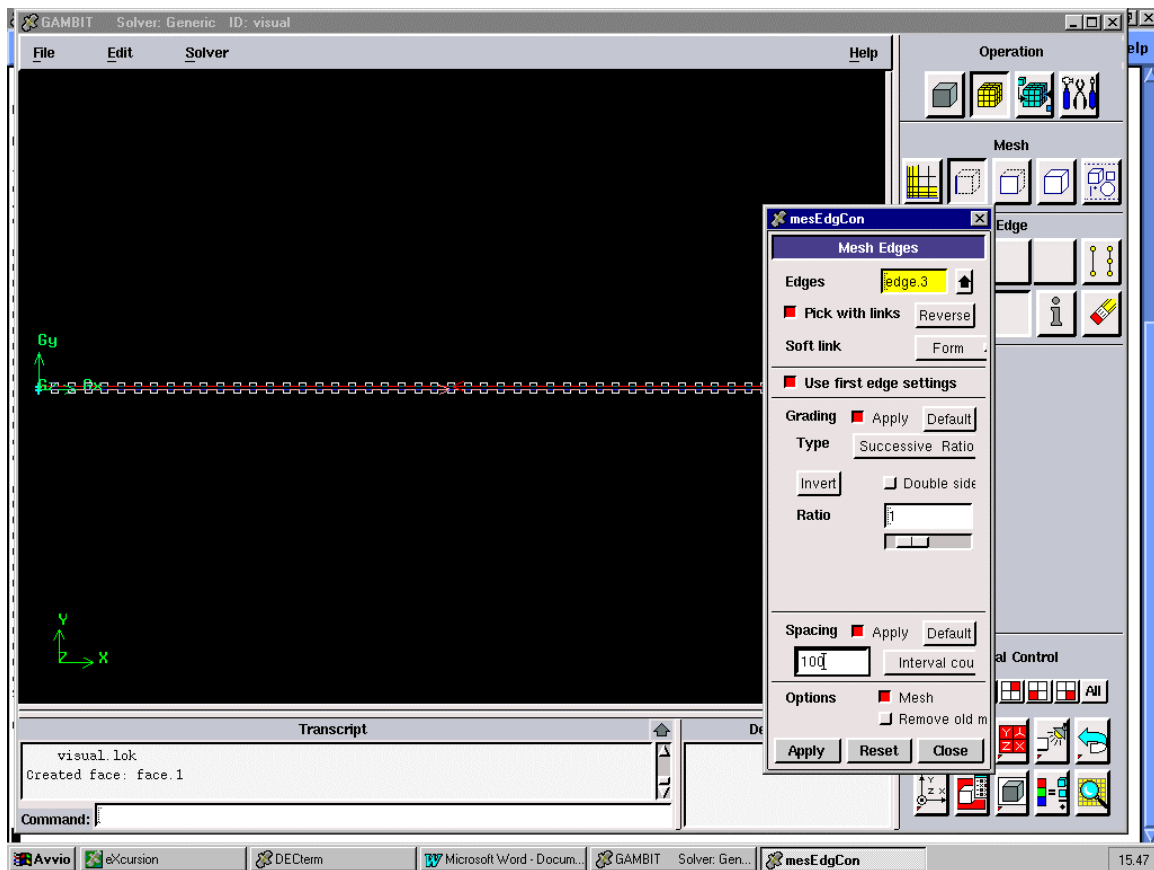


Fig.1.7 Finestra che compare premendo il pulsante *mesh edges*

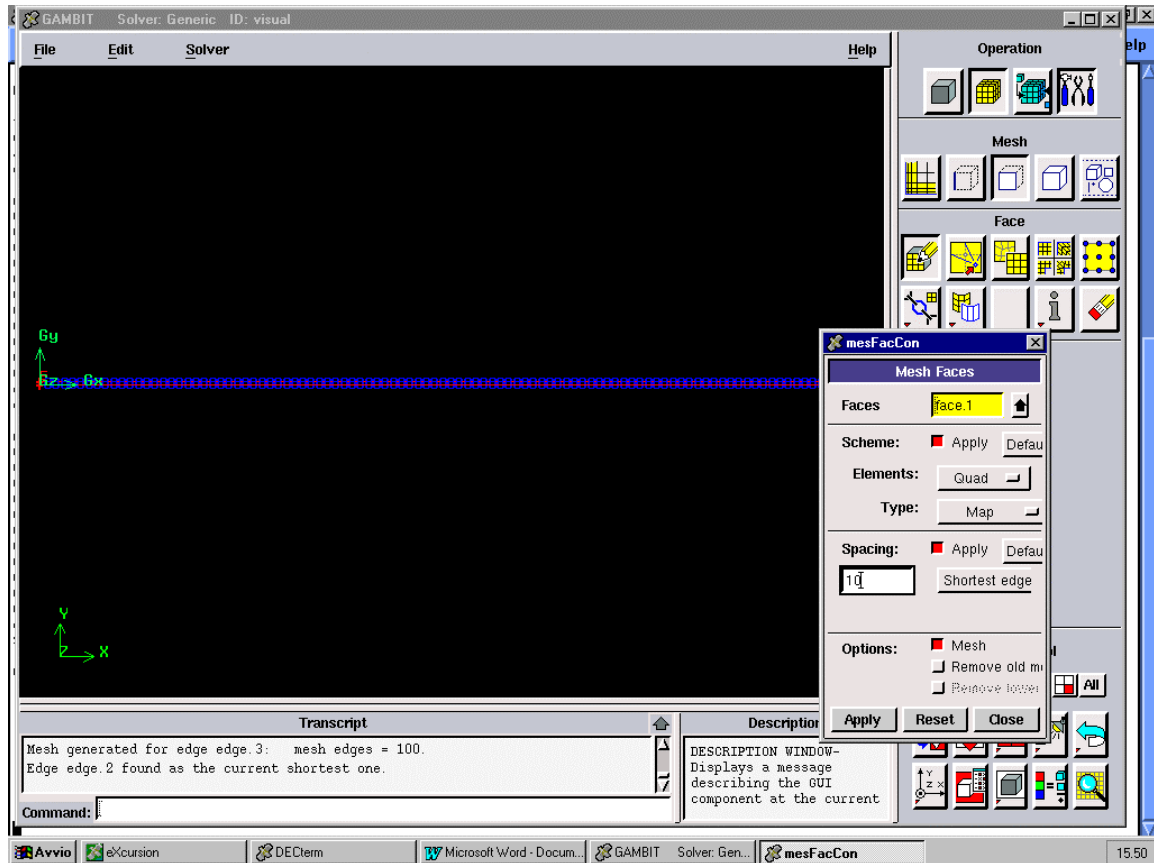


Fig.1.8 Finestra che compare premendo il pulsante *mesh faces*

Nella finestra di fig. 1.8 è necessario indicare la superficie su cui operare e, come per la *mesh* lineare, il criterio di distanziamento degli elementi e il tipo di *mesh* che in questo caso è quella quadrangolare o *quad* con elementi di tipo *map* ; è stata poi impostata l'opzione *shortest edge%* con il valore 10 per il distanziamento degli elementi in modo da generare in direzione radiale elementi quadrangolari, il cui lato è il 10% del segmento più piccolo del contorno del rettangolo, che è il suo lato minore.

La *mesh* costruita ha quindi 100 elementi lungo la direzione assiale e 10 lungo quella radiale ed è uniforme così come indicato in fig. 1.9.

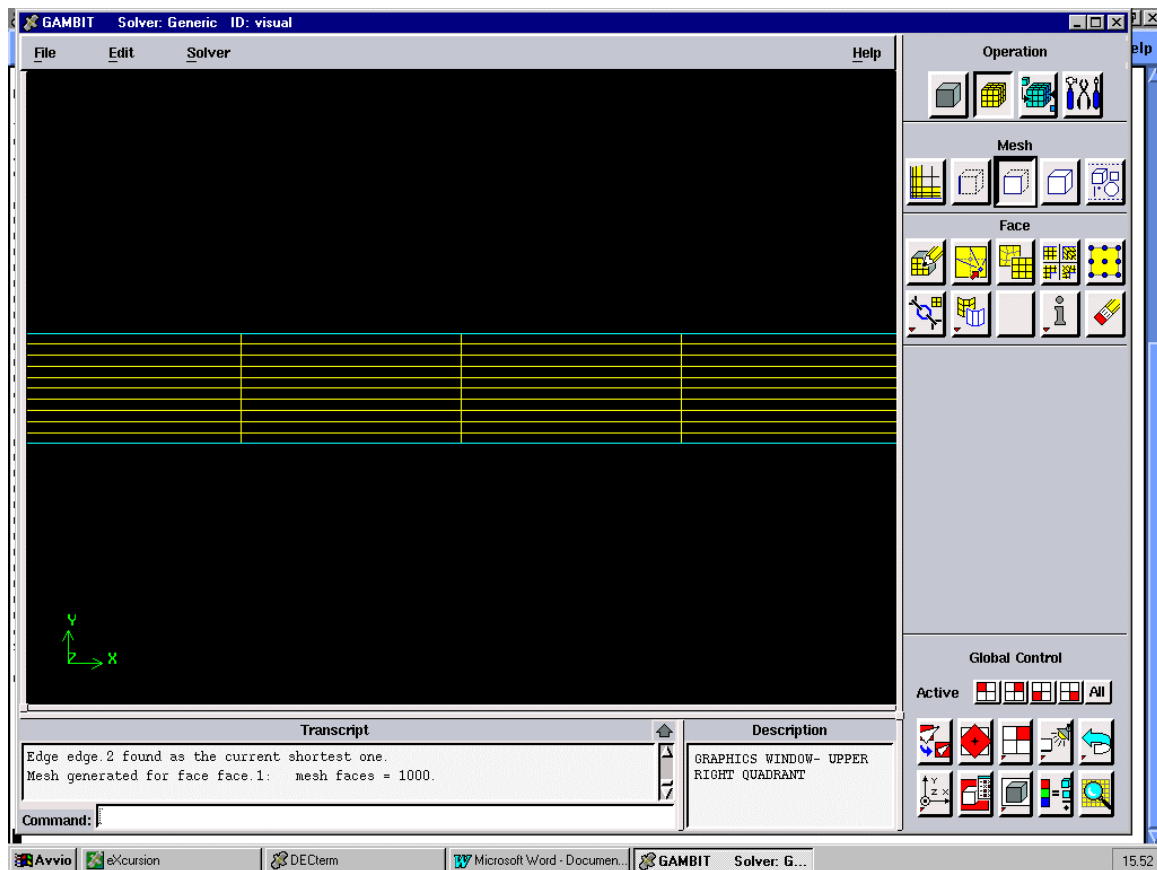


Fig.1.9 Mesh generata da Gambit in seguito alla digitazione del comando *Mesh Faces*

Ora bisogna definire il solutore con cui risolvere il modello e impostare le condizioni al contorno. Andando nel menù *solver* bisogna selezionare *Fluent 5* (fig. 1.10).

Scegliendo il solutore prima di impostare le condizioni al contorno, si ha la possibilità di definire il tipo di condizioni al contorno che può utilizzare Fluent. Per impostare le condizioni al contorno bisogna puntare il riquadro che affianca quello relativo alla creazione della *mesh* sotto *operation* in modo da far comparire la relativa finestra (fig. 1.11).