In [7]:
```
from datetime import datetime
data_ora_correnti = datetime.now()
data_ora_formattate = data_ora_correnti.strftime("%Y-%m-%d %H:%M:%S")
autore = 'Federico Targa'
print('-----------------------------------')
print("| Data e ora:", data_ora_formattate,'|')
print('-----------------------------------')
print('-----------------------------------')
print('     |Autore: ', autore                ,'|')
print('-----------------------------------')
```

```
-----------------------------------
| Data e ora: 2023-06-10 16:55:23 |
-----------------------------------
-----------------------------------
     |Autore:  Federico Targa |
-----------------------------------
```

ABSTRACT

L'equazione differenziale del trasporto, anche nota come equazione di advection o convezione, è un'equazione differenziale alle derivate parziali che descrive il trasporto di una quantità attraverso un flusso. La forma generale dell'equazione del trasporto in una dimensione spaziale è:

$\partial u/\partial t + c\ \partial u/\partial x = 0$

dove u è la quantità trasportata e c è la velocità del flusso.

In generale, l'equazione del trasporto non è risolvibile analiticamente per tutti i casi. Tuttavia, esistono alcune situazioni in cui è possibile ottenere soluzioni analitiche semplici. Ad esempio, se la velocità di flusso c è costante nel tempo e nello spazio, l'equazione del trasporto può essere risolta analiticamente utilizzando metodi di separazione delle variabili o trasformate integrali.

Alcuni esempi comuni di soluzioni analitiche per l'equazione del trasporto includono le onde progressive e le soluzioni con profili di discontinuità. Tuttavia, la complessità dell'equazione del trasporto aumenta se si considerano flussi non costanti nel tempo o nello spazio, flussi multidimensionali o condizioni iniziali/ai bordi complesse. In questi casi, spesso è necessario ricorrere a metodi numerici per approssimare la soluzione dell'equazione del trasporto.

In conclusione, mentre ci sono casi in cui l'equazione differenziale del trasporto può essere risolta analiticamente essa richiede, in generale, l'uso di metodi numerici per ottenere soluzioni approssimate.

Come detto, esistono sono diversi metodi numerici per approssimare la soluzione dell'equazione differenziale del trasporto. La scelta del metodo dipende dalla complessità del problema e dai requisiti specifici dell'applicazione. Ecco alcuni dei metodi più utilizzati:

1) Metodo delle differenze finite: Questo metodo approssima le derivate spaziali e temporali dell'equazione del trasporto utilizzando differenze finite. Esistono diverse varianti del metodo delle differenze finite, come il metodo esplicito (come lo schema di Lax-Friedrichs o il metodo di upwind) e il metodo implicito (come lo schema di Crank-Nicolson). Questi metodi possono essere applicati a problemi unidimensionali e multidimensionali.

2) Metodo delle caratteristiche: Questo metodo sfrutta le curve caratteristiche dell'equazione del trasporto per trasformare l'equazione in un sistema di equazioni differenziali ordinarie. La soluzione può quindi essere approssimata integrando lungo le caratteristiche. Il metodo delle caratteristiche è particolarmente utile per le equazioni del trasporto con flussi variabili nel tempo o nello spazio.

3) Metodo degli elementi finiti: Questo metodo suddivide il dominio in elementi finiti e approssima la soluzione dell'equazione del trasporto all'interno di ciascun elemento utilizzando funzioni di base. L'equazione del trasporto viene quindi riformulata come un sistema di equazioni algebriche che possono essere risolte numericamente. Il metodo degli elementi finiti è flessibile e può essere applicato a problemi con geometrie complesse o condizioni ai bordi non standard.

4) Metodo delle differenze spettrali: Questo metodo combina le differenze finite con l'approccio spettrale per ottenere una maggiore precisione nella soluzione. Utilizza una serie di funzioni base (come le funzioni di Chebyshev o di Fourier) per approssimare la soluzione dell'equazione del trasporto. Il metodo delle differenze spettrali è particolarmente adatto per problemi con soluzioni lisce.

Questi sono solo alcuni dei metodi più comuni utilizzati per approssimare la soluzione dell'equazione differenziale del trasporto. La scelta del metodo dipende dal problema specifico e dalle sue caratteristiche. In alcuni casi, potrebbe essere necessario utilizzare una combinazione di metodi o approcci più avanzati, come i metodi a volumi finiti o i metodi di elementi discreti.

In questa sede, ai fini di determinare una soluzione approssimata, utilizzeremo sia il metodo delle differenze finite, sia il metodo delle caratteristiche. In particolare, semplificheremo il problma considerando consideriamo l'equazione del trasporto unidimensionale con velocità costante, ovvero tale che:

$\partial u/\partial t + c\ \partial u/\partial x = 0$

METODO DELLE DIFFERENZE FINITE

In [8]:
```
#Innanzitutto, importiamo i moduli necessari per la risoluzione del problema
import numpy as np
import matplotlib.pyplot as plt
```

In [9]:
```
# Successivamente, definiamo i parametri del problema in esame
c = 1.0  # Velocità del flusso
t_start = 0.0  # Tempo iniziale
t_end = 1.0  # Tempo finale
x_start = 0.0  # Posizione iniziale
x_end = 1.0  # Posizione finale
num_x_points = 100  # Numero di punti lungo l'asse x
num_t_points = 100  # Numero di punti temporali
```

In [10]:
```
# Definiamo ora, sia l'intervallo spaziale, sia quello temporale
dx = (x_end - x_start) / num_x_points
dt = (t_end - t_start) / num_t_points
```

In [11]:
```
# Definiamo ora la FUNZIONE di condizione iniziale
def initial_condition(x):
    return np.sin(2 * np.pi * x)
```

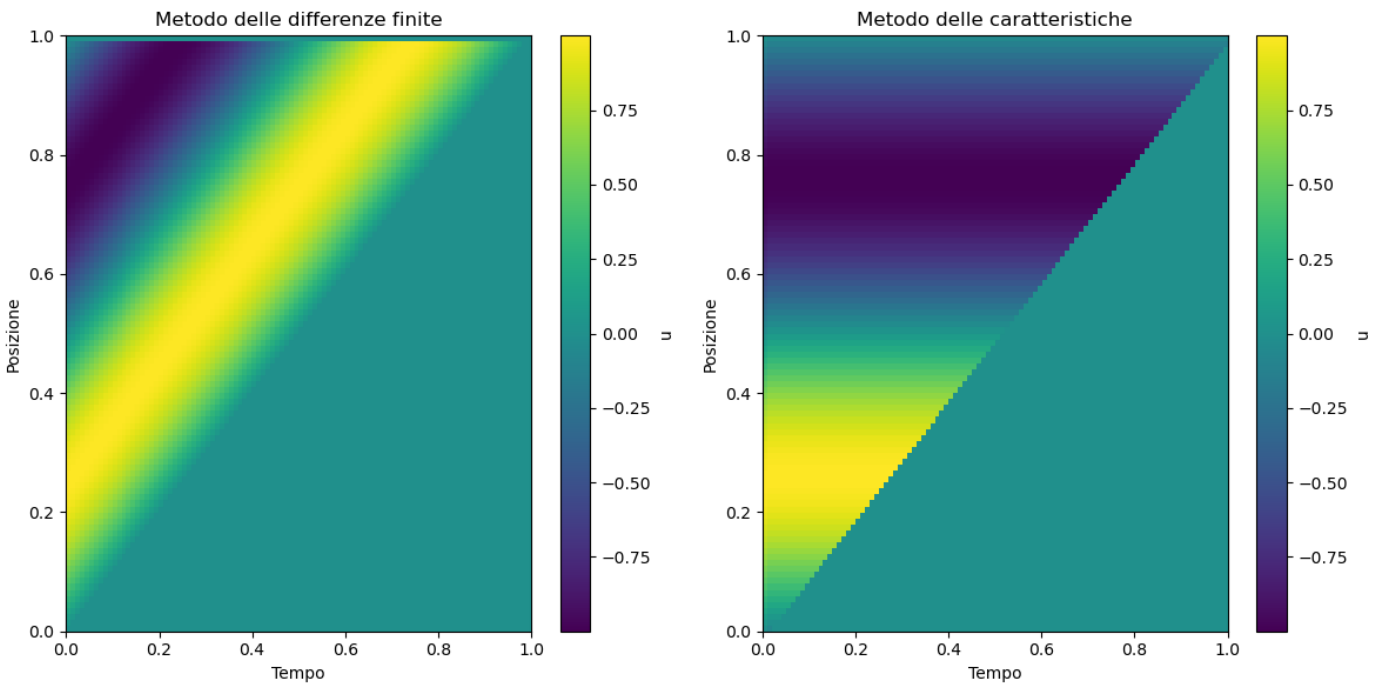
IMPLEMENTIAMO I DUE METODI

In [12]:
```
# A questo punto, definiamo una funzione che, effettivamente, implementi il metodo delle
def finite_difference_method():
    u = np.zeros((num_x_points, num_t_points))  # Matrice per la soluzione
    u[:, 0] = initial_condition(np.linspace(x_start, x_end, num_x_points))  # Condizione

    for n in range(num_t_points - 1):
        for i in range(1, num_x_points - 1):
            u[i, n+1] = u[i, n] - c * dt / dx * (u[i, n] - u[i-1, n])

    return u
```

In [13]:
```
# Allo stesso modo, definiamo una user function che implementi il metodo delle caratteri
def method_of_characteristics():
    u = np.zeros((num_x_points, num_t_points))  # Matrice per la soluzione
    x_values = np.linspace(x_start, x_end, num_x_points)  # Punti spaziali
    t_values = np.linspace(t_start, t_end, num_t_points)  # Punti temporali
    u[:, 0] = initial_condition(x_values)  # Condizione iniziale

    for n in range(num_t_points - 1):
        for i in range(num_x_points):
            x = x_values[i]
            t = t_values[n]

            if x - c * t >= x_start and x - c * t <= x_end:
                u[i, n+1] = initial_condition(x - c * dt)

    return u
```

In [14]:
```
# Valutiamo da un punto di vista numerico le soluzioni ottenute con i due metodi
u_fd = finite_difference_method()
u_mc = method_of_characteristics()
```

In [16]:
```
# Eseguiamo un plotting dei risultati
# Plot dei risultati
x_values = np.linspace(x_start, x_end, num_x_points)
t_values = np.linspace(t_start, t_end, num_t_points)

plt.figure(figsize=(12, 6))
plt.subplot(121)
plt.imshow(u_fd, origin='lower', extent=[t_start, t_end, x_start, x_end], aspect='auto')
plt.colorbar(label='u')
plt.title('Metodo delle differenze finite')
plt.xlabel('Tempo')
plt.ylabel('Posizione')
plt.subplot(122)
plt.imshow(u_mc, origin='lower', extent=[t_start, t_end, x_start, x_end], aspect='auto')
plt.colorbar(label='u')
plt.title('Metodo delle caratteristiche')
plt.xlabel('Tempo')
plt.ylabel('Posizione')
plt.tight_layout()
plt.show()
```



Ora possiamo fare una comparazione per capire quale sia il metodo più preciso. Per confrontare i due metodi e determinarne la "bontà", possiamo calcolare l'errore tra la soluzione esatta e le soluzioni approssimate utilizzando una metrica appropriata, ad esempio l'errore quadratico medio (RMSE, Root Mean Square Error); confrontando poi i due valori dell'RMSE, saremo in grado di stabilire quale dei due metodi meglio approssima la soluzione in forma chiusa.

In [18]:
```
# Calcoliamo innanzitutto l'errore quadratico medio
def calculate_rmse(u_exact, u_approx):
    return np.sqrt(np.mean((u_exact - u_approx)**2))

# valutiamo ora la sol. esatta (per confronto)
def exact_solution(x, t):
    return np.sin(2 * np.pi * (x - c * t))

u_exact = np.zeros((num_x_points, num_t_points))
for n in range(num_t_points):
    for i in range(num_x_points):
        u_exact[i, n] = exact_solution(x_values[i], t_values[n])
```

In [19]:
```
# Calcoliamo poi RMSE dei due metodi
rmse_fd = calculate_rmse(u_exact, u_fd)
rmse_mc = calculate_rmse(u_exact, u_mc)

# Stampa dei risultati
print("L'RMSE relativo al Metodo delle differenze finite è pari a: ", rmse_fd)
print('')
print("L'RMSE relativo al Metodo delle caratteristiche è pari a: ", rmse_mc)
```

```
L'RMSE relativo al Metodo delle differenze finite è pari a:  0.504900980391205

L'RMSE relativo al Metodo delle caratteristiche è pari a:  0.8631480361191325
```

Un valore RMSE più basso indica una maggiore precisione nella soluzione approssimata anche se, è molto importante tener conto del fatto che, l'errore, dipende dalla discretizzazione spaziale e temporale utilizzata.

Dunque, confrontando i valori degli RMSE ottenuti dai con i due metodi, possiamo determinare quale dei due fornisca una soluzione approssimata più accurata per l'equazione del trasporto (relativamente al caso specifico in esame)

In [20]:
```
if rmse_fd < rmse_mc:
    print("Il metodo delle differenze finite, fornisce una migliore approssimazione dell
else:
    print("Il metodo delle caratteristiche, fornisce una maggior accuratezza circa la so
```

```
Il metodo delle differenze finite, fornisce una migliore approssimazione dell'equazione
```